
Linux—Unleashing the Workstation in Your PC

Springer

New York

Berlin

Heidelberg

Barcelona

Budapest

Hong Kong

London

Milan

Paris

Santa Clara

Singapore

Tokyo

Stefan Strobel & Thomas Uhl

LINUX

Unleashing the Workstation in Your PC

2nd edition, revised and enhanced

Foreword by Jürgen Gulbins

Translation by Robert Bach & Aileen Derieg



Springer

Stefan Strobel
Schlegelstraße 19
D-74074 Heilbronn
Germany

Thomas Uhl
Obere Heerbergstraße 17
D-97078 Würzburg
Germany

The authors can be reached at the following e-mail addresses:

linux@hn-net.de
stefan.strobel@linux.org
thomas.uhl@linux.org

Library of Congress Cataloging-in-Publication Data applied for.

Printed on acid-free paper.

© 1996 Springer-Verlag New York, Inc.

Softcover reprint of the hardcover 2nd edition 1996

All rights reserved. This work may not be translated or copied in whole or in part without the written permission of the publisher (Springer-Verlag New York, Inc., 175 Fifth Avenue, New York, NY 10010, USA), except for brief excerpts in connection with reviews or scholarly analysis. Use in connection with any form of information storage and retrieval, electronic adaptation, computer software, or by similar or dissimilar methodology now known or hereafter developed is forbidden.

The use of general descriptive names, trade names, trademarks, etc., in this publication, even if the former are not especially identified, is not to be taken as a sign that such names, as understood by the Trade Marks and Merchandise Marks Act, may accordingly be used freely by anyone.

Production managed by Henry Krell; manufacturing supervised by Rhea Talbert.

Typeset in \TeX from the authors' Microsoft Word files.

9 8 7 6 5 4 3 2 1

ISBN-13: 978-0-387-94601-6 e-ISBN-13: 978-1-4684-0247-6
DOI: 10.1007/978-1-4684-0247-6

Foreword

UNIX achieved its widespread propagation, its penetration of the university domain, and its reach into research and industry due to its early dissemination by AT&T to all interested parties at almost no cost and as source code. UNIX's present functionality emanated not just from AT&T developers but also from many external developers who used the product and contributed their own further developments, which they then put at AT&T's disposal. (Consider the contributions of the University of California at Berkeley, for example.) With the rising commercialization of UNIX by AT&T (and the current owner, Novell) since 1983, and with the philosophical wars between the large UNIX vendors such as Sun, HP, Digital, IBM, SCO and the UNIX laboratory, as well the more rhetorical than factual discussions between OSF and UNIX International, such creative and cooperative continuing development became increasingly restricted, and UNIX source code today has become unaffordably expensive and de facto inaccessible.

UNIX history

Linux has changed the situation. Linux provides interested computer scientists and users with a system that revives the old UNIX tradition: Linux is available for free, and everyone is heartily invited (but not obliged) to contribute to its continuing development. When I wrote the foreword to the first edition of this book in 1994, Linux, because it ran on PC systems, had begun to penetrate the workrooms of many computer science students and computer freaks. This triumphal advance is nearly complete, and Linux is beginning to play a more serious role in commercial environments.

free & participatory

The functionality, completeness and compatibility to commercial UNIX products that Linux has meanwhile achieved is truly

Linux measures up

impressive. The rate of development is so rapid that it proves difficult to remain up to date. Today Linux can hold its own against proprietary UNIX. In fact, in certain point, such as support of PC expansion boards and peripherals, Linux has outstripped its proprietary competitors. What is of utmost importance to systems programmers and in stark contrast to commercial UNIX systems, Linux source code is freely available.

Linux overview

This book can contribute to making this rich source of software more readily accessible for users and programmers and to help readers obtain an overview. This is the task of this book, which seeks to complement the readily available Linux Manual Pages by means of an overview and numerous notes and additions. This will permit the Linux novice a more rapid initiation, better utilization of the system, and a look at areas that were unfamiliar or could not be found. Especially the instructions for installation and administration—which unfortunately vary greatly from one UNIX system to the next—will prove helpful to most Linux users.

pioneering spirit

As a sort of UNIX veteran, I am pleasantly surprised that Linux, far beyond the initial euphoria, has managed to rekindle the pioneering spirit of the early UNIX years and, for the benefit of the commercial UNIX peddlers with their often ridiculous, unresolved, and truly proprietary discussions, really shows what is possible and useful. May this book contribute to this effort.

J. Gulbins

Note from the authors

This book evolved from the experience that we gathered through various courses that we taught and from the questions that we repeatedly encounter on the subject of Linux. We have seen that novice users, in order to facilitate their getting started, frequently need a broad overview of the Linux system and its many tools rather than large amounts of precise knowledge of technical details. As such, instead of serving primarily as a reference, this book strives to impart to the reader the necessary basic knowledge that makes it possible to tap additional sources independently.

Acknowledgments

We wish to expressly convey our gratitude to the following persons, who actively contributed to the production of this book: Christian Lotz and Henner Zeller, without whose help we could never have found the time to complete this second edition; Dirk Höfle, Sascha Runge, Rainer Maurer, Maren Mecking and Roland Uhl, who contributed criticism and corrections. We owe special thanks to the director of the computing center at Heilbronn College, Dr. G. Peter, and his staff. We are obliged to J. Gulbins for contributing the foreword.

We also thank our translators, Bob Bach and Aileen Derieg, for the synergy and the brutal night shifts that they shared with us in the final stages of preparing the manuscript for production.

Contents

Introduction 1

1.1 Historical perspectives on Linux 1

1.2 Versions 4

1.3 Features 5

1.4 UNIX development and standards 6

1.5 The Free Software Foundation 7

1.6 An overview of Linux features 9

Basics 11

2.1 Multi-user operation 11

2.2 Multitasking 12

2.3 Memory management 13

2.4 Shell model 14

2.5 File systems 15

2.6 Devices 18

2.7 Shells 20

2.8 Search patterns 32

2.9 Daemons 34

2.10 Overview of commands 35

Linux Features 37

3.1 Virtual consoles 37

3.2 Linux file systems 38

3.3 Data exchange 40

3.4 Loadable Modules 43

3.5	Sound	44
3.6	Alternative shells	44
3.7	Extended commands	47
Emulators		49
4.1	DOS emulator	49
4.2	WINE	61
4.3	iBCS2 emulator	61
4.4	HP48 emulator (X48)	64
4.5	IBM 3270 emulator	67
4.6	Macintosh emulator	67
Installation		69
5.1	Linux distributions	69
5.2	Sources	72
5.3	Hardware	74
5.4	Installation	78
5.5	Creating a boot diskette	90
5.6	Boot manager	93
Configuration		99
6.1	General configuration	99
6.2	Kernel	102
6.3	Daemons	108
6.4	Serial login	114
6.5	Fax	114
6.6	Streamers and CD-ROM	116
Administration		119
7.1	The administrator	119
7.2	Booting	120
7.3	Shutdown	123
7.4	The Linux directory tree	123
7.5	Users and groups	130
7.6	Shells	132

7.7	User information	133
7.8	Backups	133
7.9	File system management	134
7.10	Upgrades	135
7.11	Boot diskettes	138

X Window System..... 143

8.1	Features	143
8.2	Structure	146
8.3	X resources	148
8.4	Window managers	148
8.5	Toolkits	150
8.6	X11 server	154
8.7	Linux as X terminal	154
8.8	X11 configuration	156
8.9	Configuration of X applications	169

Networking..... 181

9.1	Network hardware	181
9.2	TCP/IP	182
9.3	IP	184
9.4	Serial Connections	191
9.5	PPP	194
9.6	Parallel connection	199
9.7	TCP and UDP	199
9.8	Host names	200
9.9	UUCP	205
9.10	RPC	210
9.11	NIS	211
9.12	NFS	211
9.13	LAN manager	213
9.14	PC/NFS	218
9.15	Columbia Appletalk (CAP)	220
9.16	ISODE	220
9.17	Novell	220

Network Applications 223

10.1	Network daemons	223
10.2	Internet daemon (inetd)	225
10.3	Telnet	226
10.4	FTP	228
10.5	Archie	229
10.6	Berkeley r-Utilities	231
10.7	Mail	233
10.8	News	244
10.9	IRC	248
10.10	Gopher	249
10.11	World Wide Web	251
10.12	Network management	253

Support & Help 255

11.1	man, xman	255
11.2	Info	258
11.3	Newsgroups	258
11.4	FAQs and HOWTOs	260
11.5	WWW	261
11.6	Mailing lists	261
11.7	Other documents	262
11.8	Other sources	262

Applications 265

12.1	Desktop environment	265
12.2	Editors	265
12.3	Graphic programs	269
12.4	Word processing	274
12.5	Multimedia environment Andrew	276
12.6	Databases	278
12.7	Mathematical applications	280
12.8	Simulations	281
12.9	Games	283

GNU Emacs 287

13.1 Overview 287

13.2 Basic terms 288

13.3 Operation 289

13.4 Documentation and help 292

13.5 Modes 295

13.6 Packages and enhancements 296

13.7 Emacs Lisp 300

13.8 Configuration 314

Languages & Tools 325

14.1 Languages 325

14.2 C compilers 326

14.3 Pascal, Fortran, Simula, and Modula-2 327

14.4 Lisp and Prolog 328

14.5 Tcl 328

14.6 Interface builders 335

14.7 Metacard 337

14.8 awk, gawk 337

14.9 Perl 338

14.10 Editors 339

14.11 GNU Debugger (GDB) 339

14.12 Make utility 341

14.13 Imake 342

14.14 RCS 342

14.15 xwpe 343

14.16 Example 344

14.17 Porting software 348

Reference 353

Appendix 409

A.1 Overview of /etc files 409

A.2 Overview of /etc directories 411

A.3 Configuration of the kernel 412

A.4 Further reading	414
Index	417

Introduction

For some time 32-bit machines have been a hot topic in the world of PCs. It seems that more powerful operating systems will soon be displacing DOS. Meanwhile, at least in the professional literature, lively discussion has been raging about what the future standard will be. Two alternatives seem to be emerging for the domain of server operating systems: Windows NT, and UNIX variants such as Solaris 2, UnixWare, and NextStep 486. In this context OS/2 plays no significant role since it is seen more as a competitor to Windows in its current version and future 32-bit versions.

UNIX vs. Windows NT

We cannot yet predict which system will finally predominate. However, the significant rise in the power of hardware in recent years has unleashed the demand for a modern operating system that makes use of these developments. Under a modern server operating system, the borderline between classical UNIX workstations and high-end PCs will tend to become more fluid.

workstations

1.1 Historical perspectives on Linux

An extremely powerful alternative to the above proprietary systems has evolved far from all the big debates on strategy. The system is Linux, a UNIX system for Intel processors that is available for free.

Linux was developed by a young Finnish student named Linus Torvalds. His initial goal was not to develop a full-scale operating system, however. At first he only wanted to acquaint himself with and understand the special task-switching commands of the 80386 processor. To compile his test program, he used MINIX, a pedagogical operating system by Andrew Tannenbaum for teaching and learning operating systems.

386 processor

Yet, due to its didactic orientation, MINIX had some shortcomings. The ambitious student soon exhausted the possibilities of the UNIX-like system. From his test program, he began step by step to develop a small operating system (kernel) that ran in the protected mode of the 80386 and thus optimally exploited the processor.

After the task switcher, Torvalds wrote a simple keyboard driver to allow him to work interactively with the system. At this point Linux still relied on parts of the MINIX system, but that was soon to change.

To avoid having to develop a new file system as well, Torvalds decided to adopt the MINIX file system. This not only saved him a great deal of work, but also provided from the start a stable system for managing the hard disk. After a few months the developer considered the system to be mature enough to present it to a more general public.

In August 1991 the complete source code of Linux appeared for the first time on Finland's largest FTP server (Internet address `nic.funet.fi`). It was announced as a "freely distributable MINIX clone" and caught the attention of only a few interested parties on the network. Only two months later, Torvalds published the next version (0.02), which contained some rudimentary UNIX commands. The accompanying GNU compiler (`gcc`) permitted the compilation of small C programs and thus enabled the porting of a UNIX shell (`bash`).

The early decision to adhere to POSIX, a family of standards of the Institute of Electrical and Electronics Engineers (IEEE), played a deciding role in ensuring the portability of standard UNIX software to today's Linux. However, it took until the end of the year before Linux received more widespread notice. The breakthrough came on January 5, 1992, with version 0.12. Linux had attained sufficient power to interest a larger community of developers. The system had meanwhile acquired a swapping mechanism that gave it an unequivocal edge over MINIX.

Over time an ever-increasing number of interested developers were sending corrections and suggestions for improvements to Finland and thus participating in the improvement of the Linux system. Early developments contributed in this way include the

POSIX Job Control in version 0.12 and the switchable virtual consoles.

The Internet proved to be an important tool for the rapid development of Linux. The Internet is a wide-area network (WAN), an information highway connecting more than six million computers and allowing the fast exchange of all kinds of information. The Internet permits Linux developers to exchange comments, improvements, and programs.

Internet

Early on, Torvalds was bombarded with over 60 e-mail messages per day, which he could hardly read and answer. Only after several discussion groups for Linux were set up did the flood of mail subside. Today there are several newsgroups concerned with Linux. The most important is `comp.os.linux.announce` (c.o.l.a.), where new developments and program versions are announced. Mailing lists were set up for Linux developers to permit a similar kind of information exchange

discussion groups

mailing lists

In addition to letters and information, files can be exchanged via the Internet, which makes it possible to organize the distributed development of larger software systems, as Linux so impressively demonstrates.

The rapid flow of information proves to be a tangible advantage not only for developers but also for users of Linux. If the user encounters any problems during installation or detects errors during operation, then with a little luck and the Internet, an adequate solution is only a couple of hours away. Even commercial service contracts seldom provide such extensive support.

support

Naturally, not every Linux user has access to the Internet, but even then the user is not deserted. Many mailbox networks have set up Linux discussion groups, so that a modem suffices for keeping on top of things.

mailboxes

An interesting aspect of Linux history is that there was never a strict hierarchy or authority that managed the development in any way. Rather, the project has been fueled by the enthusiasm of many individual Internetters who continue to contribute new improvements and suggestions. These are often professional developers or employees of large institutions who contribute their free time.

participatory
development

Although Linus Torvalds continues to handle the concrete further development of the kernel, quite a few competent allies have taken over other areas of the system. Such areas include the porting and maintenance of the GNU C compiler and the C libraries for Linux, the maintenance and adaptation of the X Window System, and networking. Other Linux devotees are working on user and system documentation, or assembling an installable system on diskettes or CD-ROM.

Free source code is available not only for the kernel but also for most application programs. They come primarily from the huge UNIX freeware archives on the Internet. Periodically via the Internet a software catalog (Linux Software Map) is distributed; it currently contains some 1300 software packages. There is scarcely a domain for which some suitable software cannot be found.

Since much development at American universities is carried out with UNIX, and such developments become public domain, many implementations in the area of research are available for Linux. One example is compilers for both well-known and more obscure programming languages. Also, the database systems Ingres and Postgres from the University of California at Berkeley have been ported to Linux.

Although the emphasis remains on freeware, commercial applications are also available, such as a Modula-2 compiler, a Smalltalk development system, an interface builder, CAD software, several database systems, and the OSF/Motif graphical user interface that has become a standard in the UNIX world.

1.2 Versions

The further development of Linux is currently experiencing big leaps similar to the first implementation of the kernel. Version 1.0 was scheduled for December 1992 but was delayed—not because of a lack of stability, but because the functionality had not yet matched that of proprietary UNIX systems.

Thus the release of Version 1.0 was repeatedly delayed. In retrospect, Torvalds says that he should have declared the first stable and usable version 0.12 as version 1.0, for the zero in the version

number apparently scared off many potentially interested parties from delving deeper into the system.

The final version 1.0 was released in March 1994, and developments continued with the version numbers 1.1x. This system of numbering led to a great deal of confusion among users and interested parties. Various CD producers have also contributed to the confusion by selling several Linux distributions with their own version numbers. Linux itself, however, is only a small component that easily fits on a disk several times when compiled. Asking about the Linux version on a CD is thus misleading. Instead, you should ask about the version of the kernel, the C library, the compiler, or X11.

distributions

Another popular misconception is that the higher the version number, the better or more stable the software is. This is not the case, however. For a long time, the Kernel 1.0 Patchlevel 9 (or 1.0.9 for short) was the only stable kernel; the kernels with 1.1.x numbers still contained many new and not yet completely polished functions. They were intended for developers and often changed several times in a single week.

This process ended with the release of version 1.2. Linux 1.2 contains drivers for the NCR SCSI chipset used in many PCI motherboards; it is a prerequisite for the latest versions of the DOS emulators and the iBCS2 emulation.

Linux 1.2

The situation is similar with the versions of the GNU C compiler: version 2.5.8 was by far more stable than version 2.6.0. So it is not possible to determine the quality of a Linux distribution solely by the numerical value of the version number. We recommend the Slackware or the Linux Universe distributions.

GNU C

The current version of Linux has all the important features of its commercial competitors, and due to its efficient design, it extracts much higher performance from a given hardware configuration. This applies to the graphical user interface as well as to the kernel.

maturity

1.3 Features

An option that may prove particularly interesting for use in commercial areas is the possibility of running programs for other

COFF, ELF	PC-based UNIX variations in COFF or ELF format under Linux.
iBCS2	The Linux user has access to a practically unlimited supply of professional applications with the iBCS2 emulator that was specifically developed for this purpose. DOS programs also run with a DOS emulator. An emulator to directly execute MS-Windows programs under X11 is not yet entirely ready, but it does already promise some interesting perspectives for the future.
DOS	
X Window System	Contrary to many other UNIX systems, Linux already employs the newest version of the X Window System, X11R6. Additional Linux features that proprietary UNIX systems seldom provide include the support of INMOS transputer boards and the option to run TCP/IP via the serial or parallel port. Direct kernel support for ISDN boards for fast network connections over long distances makes Linux interesting for communication tasks. However, since there is no roadmap for the further development of Linux, the system will surely provide some surprises along the way.
pronunciation	Even insiders often mispronounce the word “Linux”. Many users consider it an American term and pronounce it with an English long i and short u. The Finnish pronunciation is the correct one, amounting to “lee-nooks” for an English speaker.

1.4 UNIX development and standards

Ritchie & Thompson	The history of UNIX dates back well into the 1970s. In 1971 Dennis Ritchie and Ken Thompson at AT&T’s Bell Laboratories developed the first version. With the availability in 1973 of the compiler
C	language C, which had evolved from BCPL and B, most of the UNIX system was rewritten, which later proved to be a great advantage for porting the system to other processors.
AT&T	Due to an agreement with the U.S. government, AT&T could not market its quite successful system. Therefore, AT&T gave UNIX as source code, although without support, to universities, where its popularity grew. With Version 7 in 1979, AT&T announced a change in its licensing policy: UNIX source code would only be provided for a fee. This prompted the University of California
BSD UNIX	at Berkeley to develop its own variant, BSD (Berkeley Software Distribution) UNIX. In 1983 AT&T announced the marketing of its

enhanced System V. The System V Interface Definition specified the programming interface to this system.

System V

Companies like Sun Microsystems, Microsoft, and DEC developed their own versions of UNIX (SunOS, Xenix, ULTRIX), which in time unnecessarily encumbered the porting of software between these systems. In order to merge the two main branches of UNIX (BSD and System V), in 1990 AT&T propagated its Release 4 of System V as a new standard that encompasses all previous variants of UNIX.

System V Release 4

Other institutions also have recognized the need for a standardization of UNIX. The Institute of Electrical and Electronic Engineers (IEEE) developed the POSIX standard for UNIX-related operating systems. This standard is divided into several parts. POSIX 1003.1 describes only the lowest-level system interface; 1003.2 will define a standard for shells and commands; 1003.7 covers the possibilities of system administration. Although POSIX is actually based on the UNIX system interface, this standard will also be supported by other operating systems (e.g., Windows NT).

standardization

POSIX

A body consisting primarily of UNIX manufacturers has released another standard. Although the X/Open Portability Guide is based on POSIX 1003.1, it provides extensions in certain points. Within the realm of the COSE Initiative (Common Open Software Environment), the importance of the X/Open Consortium rose significantly. The present goal is to release a uniform desktop-user interface, the Common Desktop Environments (CDE) and programmer interface for all available UNIX variants. This should drastically reduce the effort required for porting software. Linux adheres to the POSIX standard.

X/Open

CDE

1.5 The Free Software Foundation

In addition to its orientation to the POSIX standard, Linux is also largely subject to the General Public License (GPL) of the Free Software Foundation (FSF). FSF was founded about a decade ago by Richard Stallman, the developer of the legendary GNU Emacs editor. The organization “aims to make high-quality free software

General Public License

Richard Stallman
free software

available to everyone.” Note that *free* in the title of the organization refers to “freedom”, not “zero dollars.”

This kind of freedom means that copying and distributing software, including the source codes, is not to be restricted. This makes free software fundamentally different from public domain software or shareware. It is protected by copyright, and the license requirements are regulated by the GPL.

commercial distribution

Software that is subject to the GPL may also be commercially distributed, but it must be possible for anyone to copy it and pass it on. The source code must be included. If developers use free software as the basis for their own developments, then this development must also be made available under the GPL. This does not apply to software that was compiled with the GNU C compiler or edited with the GNU Emacs editor, but rather to programs using a source code that is subject to the GPL.

quality

A frequent result of this practice is an increase in the quality of software, from which everyone benefits. For instance, the Next Company used the GNU C compiler as the basis of its Objective C compiler. So what was available to them was a relatively mature and freely accessible compiler. Under the GPL, the new additions were made available to the general public, so now the GNU C compiler works not only with ANSI C and C++, but also with Objective C.

Objective-C

GNU

The GNU project represents FSF’s attempt to develop a complete operating system that can be freely copied and is largely compatible with UNIX. GNU, by the way, stands for “Gnu’s not UNIX.” In addition to the GNU C compiler and the Emacs editor, numerous UNIX-compatible commands and tools were developed in the course of this project that are currently used in almost all Linux distributions. What the FSF and GNU project have been lacking is an operating systems core. Although the GNU kernel (Hurd) was already in the works before Linux emerged, it cannot be employed yet by users. Hurd is based on the Mach-3 microkernel, and someday it may well be technologically superior to Linux.

Hurd

Since quite a few UNIX commands and utilities originated in the GNU project or are at least subject to the GPL, Linux has profited from the project. At the same time, Linux fulfills the aim of the GNU project, as the Linux kernel, together with the FSF tools and other

Linux & FSF

freely accessible utilities, represents a complete UNIX system that is free of charge.

Currently, the further development of such essential elements as the C compiler and the C library is done in collaboration and is coordinated by GNU and Linux developers. Popular FTP servers now offer the interested programmer an overwhelming multitude of software that is subject to the GPL. In addition to programming languages such as C, C++, Smalltalk, Lisp, and Fortran, there are also various editors, debuggers (gdb), and even a PostScript interpreter (Ghostscript).

further developments

FTP server

1.6 An overview of Linux features

To give the reader a better orientation, we offer the following summary of Linux's most important features:

- **A full-fledged, 32-bit multi-user/multitasking UNIX system.** Linux permits multiple users to execute (different) programs simultaneously and thereby fully exploits the capacity of the Intel 80386 processor and its successors. The resulting performance is definitely comparable to a classical RISC workstation.
- **Orientation to common UNIX standards (POSIX).** Because Linux adheres to existing standards for UNIX, available software usually can be ported to Linux without problems.
- **Network support (TCP/IP).** A Linux machine can easily be integrated into a TCP/IP network. Linux supports common Ethernet boards and TCP/IP connection via modem (SLIP/PPP) and ISDN.
- **Graphical user interface (X11).** The Linux system includes the current version (Release 6) of the X Window System. OSF/Motif, the standard user interface for proprietary UNIX systems, is also available.
- **GNU utilities and programs.** Many of Linux's commands and utilities emanate from the GNU project and contribute much functional enhancement.

multi-user

multitasking

performance

standards

network

X11R6

OSF/Motif

GNU

- compatibility • **Complete UNIX development environment.** Linux permits the development of programs that run problem-free on other UNIX systems. In addition to the GNU C/C++/ Objective C compiler, numerous editors, and several version control systems, there are many other software development tools.

Basics

In order to understand the following chapters, the reader will need some knowledge of computer science in general and UNIX in particular. To ease the transition into this material for readers who are newcomers to UNIX, we present some of the most important concepts and terms in this chapter.

2.1 Multi-user operation

In classical data processing, a central mainframe handles all required DP tasks. Serial connections link terminals (simple text-oriented consoles with keyboards) to this mainframe. Many users sharing a single mainframe necessitates a system of access control and user management to achieve fair distribution of the shared resources.

mainframe
terminals

Unique user names, also known as *user IDs*, form the basis for such a system. Every user is assigned such a user ID (identification); the user must provide the system with this user ID and its associated *password*, a procedure known as *logging in* to the system. On the basis of the user ID and its associated *access privileges*, or *permissions*, the system appropriates access to files and other resources.

Systems that provide such management for multiple users are known as multi-user systems. UNIX is a typical multi-user system, providing for each user a user ID and one or more group IDs. A user can belong simultaneously to multiple groups. This is practical if the user is involved in multiple projects or needs to have access to data in different domains. Permissions for directories and files can be assigned individually for users and groups.

multi-user

All multi-user systems have one privileged user who administers the system. This user is called the system administrator, or

system administrator

root superuser. Under UNIX this privileged user has the name `root` and the numerical ID 0. This user's permissions are unlimited. The administrator can create new users and assign permissions. Chapter 7, "Administration," explains the tasks of the system administrator in more detail.

from central
mainframe to network
multiple terminals

In recent years hardware prices have fallen while performance has risen. The possibilities afforded by decentralized systems with their graphical user interfaces led to a decline in the acceptance of mainframes. Instead of a central mainframe with many terminals and numerous users, today's trends reveal an increasing number of workplaces where each user has sole use of one or more computers. These computers are connected via a network to each other and to other computers and servers to permit the easy exchange of data.

UNIX workstation

A similar development has occurred in the area of UNIX: the purely text-oriented UNIX system has evolved into a graphical desktop UNIX workstation. The trend that a single user might use multiple computers at the same time in order to simultaneously access various programs and data is also reflected in *virtual terminals*. This feature allows a user to log in to the same computer repeatedly, even though the user physically only has one monitor; a special key combination makes it possible to switch between various virtual terminals.

2.2 Multitasking

many tasks
simultaneously

Newer multi-user systems are usually multitasking systems as well. They are capable of processing many tasks quasi-simultaneously, which simple multi-user systems cannot necessarily do.

processes
and daemons

The smallest unit that can be handled in parallel in such a system is called a *process*, or *task*. In UNIX, which is both a multi-user and a multitasking system, we generally speak of processes. Processes that execute in parallel on a UNIX system could be programs from different users or programs that always run in the background (daemons).

IPC A significant characteristic of modern multitasking systems is the availability of interprocess communication (IPC). This

encompasses functions for synchronization and data exchange between processes.

On conventional computers with only one processor, the CPU must be allocated alternately among the individual processes in order to give the user the impression of simultaneous execution. This task is managed by the scheduler, a special process that maintains a list of the normal processes and sees to it that the processor handles the next process at certain time intervals.

scheduler

There are various strategies that a scheduler can use to determine which process to handle next. One very simple strategy (round robin) selects the next respective process in the list at regular intervals (e.g., 50 ms) and puts it at the end of the list after the allocated time if the process is not yet finished. Another strategy assigns each process a priority, whereby processes with higher priority are allocated more CPU time.

UNIX employs *nice levels*, which allow the user to influence the internal priorities of processes. This allows the user to reduce significantly the encumbering of the system by programs running in the background. Likewise, the system administrator can also raise the priority of important processes to ensure faster execution.

nice level

2.3 Memory management

Memory management in today's UNIX systems differs substantially from that of a simpler operating system. For instance, a UNIX operating system may pretend to provide more main memory to the programs than is actually available.

The method used to implement virtual memory management in Linux is called *paging*. With the help of tables, the operating system maps a large logical address space onto a smaller physical address space. When processes demand more main memory than is physically present, individual segments of logical memory that have not been referenced recently are relocated onto the hard disk.

paging

When a program accesses a logical address that is currently located on the hard disk, the respective memory segment (called a *page*) is loaded into main memory, while another memory segment must be written to the hard disk to compensate. Due to the

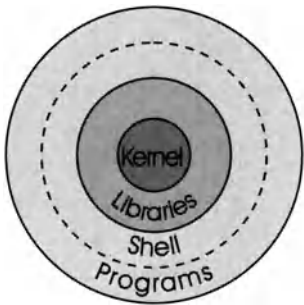


Figure 2.1. Schematic structure of a UNIX system.

significantly higher access time of a hard disk compared to main memory, there is naturally a price to be paid in terms of execution speed.

In order to be able to use the hard disk for virtual memory management and the logical main memory, *swap files* or *swap partitions* must be created on the hard disk. Without such partitions or files, main memory is limited to its actually available physical size.

2.4 Shell model

The structure of a UNIX system is often depicted as a *shell model* (see Figure 2.1). The kernel of a UNIX system contains components like the *scheduler* and the *device drivers*. These routines permit access to the interface hardware and external devices. The memory management also resides in the kernel.

The processes in the kernel are different from the processes running in the shell around the kernel. User processes can be interrupted at any time; they are subject to the control of the scheduler; and a certain region of memory is allocated to each of them. If a user process attempts to access a region of memory outside its own, the process is aborted with the error message “segmentation fault.” The contents of the current memory for the process may then be written into a file called `core` (a core dump). This file can be useful for the developer looking for errors.

By contrast, kernel processes have unrestricted access to all resources of the computer. Hence we speak of two different modes in which processes can run: the user mode and the kernel mode. The outer shell of a UNIX system consists of programs that directly make contact with the user. This outer shell includes the command shell, which executes operating system commands, as well as application programs such as word processing and databases.

user mode/kernel mode

Between the outer shell and the kernel are the various libraries that provide access to their library functions (usually written in C) and to kernel routines. These libraries are normally linked to a program after its compilation, thus adding the library routines to those of the program itself.

libraries

links

Since statically linked programs demand a large amount of memory, modern programmers usually employ *shared libraries*, which consist of two parts. A small part containing only references to the library is linked to the program. The library itself is actually loaded only when the program is executed. This allows multiple programs to use the routines in shared libraries simultaneously, which also saves memory.

shared libraries

Another advantage is the possibility to exchange a shared library for a newer version without having to relink the programs relying on it. However, this assumes that the routines in the new library are invocation-compatible with those of the old version.

version upgrade

2.5 File systems

A file system manages data stored on a hard disk. Although every computer system has such mechanisms, they can differ substantially. Modern file systems have a hierarchical structure (see Figure 2.2). The user can distribute files in various directories, making it easier to maintain an overview. The user accesses the files via the path. Contrary to DOS, UNIX uses the slash (/) as the delimiter with a path.

hard disk

paths

Under UNIX, paths can be specified as absolutely (with a / at the beginning) or relatively to the current directory. A user's home directory is particularly important here. Personal data is stored in this directory, and this is where the user lands after logging in.

home directory

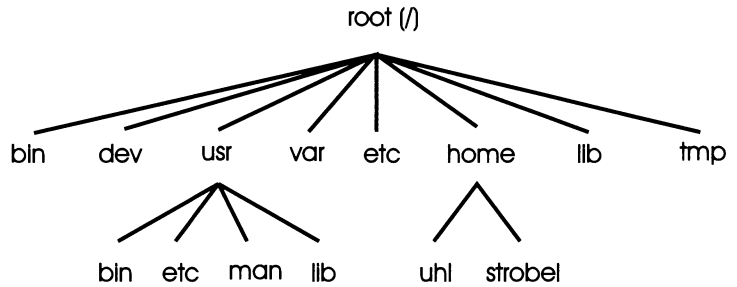


Figure 2.2. Excerpt from a UNIX file tree.

no drive letters
disks

DOS addresses disk drives and the partitions of a hard disk with a drive letter. UNIX merges all of these to a single file system and does not assign them separate designations. This means that the user can no longer distinguish the individual drives and partitions. Only one large drive with a single file system appears to exist. Problems arise in the management of diskettes and other removable storage media since these are not permanently in the drive. Before access to a removable storage medium is possible, it must be linked to the system using the `mount` command, which normally only the system administrator can do.

FAT

The management of files and free blocks differs considerably under DOS and UNIX. DOS creates a File Allocation Table (FAT) on each drive to record the free and allocated sectors. Another sector contains the root directory. Besides the names of the contained files, a DOS directory also contains their attributes, such as size and date.

i-node

By contrast, UNIX allocates an i-node for each file, in which the most important attributes are stored, such as name, permissions, and start block. Directories thus contain only references to the respective i-nodes. Because it is both more economical in terms of storage space and more efficient regarding access, such a structure proves better suited to the management of larger file systems than a FAT system.

Permissions

owner

When a file is created under UNIX, the operating system stores not only the file name and the date of creation but also the user ID of its creator (or owner) as well as the group that owns the file. To be

The diagram shows the output of the `ls` command for a file named `demo.txt`. Each field is labeled with a bracket and a title above it:

- `-`: file type
- `rw`: owner
- `r`: group
- `--`: others
- `2`: link counter
- `tu`: name of owner
- `l`: name of group
- `12354`: size
- `Aug 21`: date
- `18:03`: time
- `demo.txt`: file name

The permissions `-rw-r--r--` are grouped under a bracket labeled "permissions".

Figure 2.3. Displaying permissions using the command `ls`.

able to protect the files of a file system against undesired access, the permissions for each file are stored separately. Thus access to a file can be limited to its owner or to a certain user group. It is also possible to define general permissions. We further distinguish among read, write, and execute permissions, which are reflected in the output of the `ls` (list) command by the letters `r`, `w`, and `x`. The position of the letters in the output indicates whether the permissions apply to the owner of the file, the group owners of the file, or everyone else.

access

read, write and execute

Subdirectories are an exception. Read permission suffices only to read their contents. In order to change to a subdirectory, however, a user must possess both read and execute permissions for that subdirectory.

directories

Links

Another feature of the UNIX file systems is the possibility to create links. If access to a file is to take place from various points in the file systems, this file could simply be copied. Naturally this approach would mean a waste of storage. In such cases under UNIX, the creation of a *link* proves a more practical alternative.

references

UNIX links to a file can be either hard or symbolic links. A hard link is an additional reference from a directory to a file or its i-node. The link counter tabulates the number of such references. If a file to which several links refer is to be deleted, then the link counter is decremented by one. Only after the link counter reaches the value one can the file be deleted physically.

hard link

link counter

Since the numbers of i-nodes are unambiguous only within a file system, hard links cannot be created for files intended for use outside a given file system. By contrast, symbolic links can reference

symbolic link

any directory entries (subdirectories or files). Whether the referenced file actually exists plays no role in the creation of the symbolic link. The output of the `ls` command indicates the difference between these two kinds of links:

```
linux1:/etc>ls -l hosts passwd
lrwxrwxrwx 1 tui  users   10 Aug 21 18:05 hosts -> /etc/hosts
-rw-r--r-- 2 root  root   863 Aug  9 15:00 passwd
linux1:/etc>
```

Symbolic links are represented with an `l` (el for link) in the file types column for the file type and a visible reference to the original file. Hard links can only be recognized by the increased link counter in the second column (after the permissions).

Virtual file system

In order to support the development of different file systems, Linux provides an additional layer, the *virtual file system*, between the kernel and the actual routines of a file system s, as in proprietary UNIX systems. The virtual file system defines a number of routines that must be available in every file system, for opening, reading, writing, and closing files. This unambiguous interface enables the problem-free coexistence of different file systems.

2.6 Devices

UNIX maps hard disks as well as terminals and other devices onto special files in the directory `/dev` of the file systems. Thus the programmer can access such devices in the same way as normal files.

This file-like view of devices also has advantages for the user. If the user wants to output a file to the printer instead of to the console, then it suffices to redirect standard output to the respective device (`/dev/lp1`):

```
linux2:/home> cat outut.txt >/dev/lp1
```

/dev/console	system console
/dev/mouse	serial mouse
/dev/hda	first AT-bus hard disk
/dev/hda1	first partition of first AT-bus hard disk
/dev/hda2	second partition of first AT-bus hard disk
/dev/hdb	second AT-bus hard disk
/dev/hdb1	first partition of second AT-bus hard disk
/dev/sda	first SCSI hard disk
/dev/sdb	second SCSI hard disk
/dev/lp0	first printer port (LPT1)
/dev/null	null device (all output is suppressed)
/dev/ttyN	virtual console
/dev/ptyN	pseudoterminal for login from network
/dev/ttySN	serial port

Figure 2.4. List of the most important device drivers

Likewise the floppy disk drive (/dev/fd0), the mouse (/dev/mouse), and the hard disk are addressed via an entry in the /dev directory (see Figure 2.4). Files in directory /dev and the respective devices are associated via two numbers, the major device number and the minor device number. These also form the interface to the kernel. In addition, there are two kinds of devices, *character* and *block* devices. The former are character-oriented and are used primarily for devices like terminals and serial ports. The latter are used for transferring large data blocks for hard disks and other data storage devices. For block devices, the access position can be changed (seek). The major and minor device numbers are evident in an `ls -l` output.

mouse
device drivers

X11 or major and
minor device numbers
character and
block devices

```
linux1:/dev>ls -l
brw-r----- 1 root    root      3,   0 Aug 29  1992 hda
brw-r----- 1 root    root      3,   1 Aug 29  1992 hda1
brw-r----- 1 root    root      3,   2 Aug 29  1992 hda2
...
crw-rw-rw-  1 root    root      4,   0 Aug 16 12:26 tty0
crw--w--w-  1 tul     users     4,   1 Aug 21 15:15 tty1
...
linux1:/dev>
```

device driver The major device number (fifth column in the above listing) specifies the type of the device. Normally each major device number has a corresponding device driver in the kernel. When multiple devices of the same type are connected, they are distinguished by their minor device number (sixth column in the above listing) and served by the same driver.

2.7 Shells

command.com The shell provides the interactive interface between the operating system and the user. Even in this era of graphical user interfaces, many UNIX users still prefer to enter commands this way.

 We can compare the tasks of the shell with those of `command.com` under DOS, although a UNIX shell affords significantly more features than the DOS command interpreter.

Standard shells

sh, ksh, csh Most proprietary UNIX systems include three kinds of shells: a Bourne shell (`sh`), a Korn shell (`ksh`), and a C shell (`csh`). The Bourne shell was the first UNIX shell; thus it affords little in the way of comfort. The Korn shell is an extension of the Bourne shell and appears relatively often with proprietary UNIX systems.

alias substitution Like BSD UNIX, the C shell was born at the University of California at Berkeley. In contrast to the Bourne and Korn shells, the C shell offers a simple history function, allowing the UNIX user to return to commands that have already been entered on the command line. *Alias substitution* makes it possible to automatically replace certain commands that were entered in an initialization file or interactively. Due to its C-like syntax, this shell variant has been a particular favorite among programmers.

C-like syntax When it is started as login shell, the C shell executes the

.login commands of the file `.login`, which must be located in the home directory for this purpose. Since a user normally possesses only one active login shell, the contained commands are executed only once at login. On the other hand, a shell script that is executed at the start of each new C shell is the file `.cshrc`, which likewise must be located in the home directory.

Enhanced shells

As a rule Linux does not employ any of the above shells. Instead, Linux uses enhanced variants of these shells. These more comfortable shell variants, which are available for free for almost all UNIX systems, render the original shells obsolete. The Bourne Again shell (`bash`) has displaced the Bourne shell `bash`, and the `tcsh` shell has supplanted the C shell. System administrators usually use a Bourne shell because most administration scripts require it. The following explanations thus refer primarily to this shell variant.

more comfortable

bash and tcsh

Interactive Use

UNIX commands such as `ls`, `cd`, and `man`, which are entered in the normal command line, are not executed by the UNIX operating system itself; instead, they are programs that are usually located in the directories `/bin` and `/usr/bin`. A user entering such commands is not in direct contact with the UNIX operating system, but only with its outermost shell, a program named `shell`.

After logging in, UNIX system users normally find themselves in an interactive shell. The shell only recognizes a few commands and spends most of the time reading user input from the standard input device, i.e., the keyboard, and starting the respective program. With the input of `ls -l`, for example, the shell invokes the program `/bin/ls` with the option `-l`, and the contents of the current directory are displayed on the console:

standard input

```
zeus:/home/uhl> ls
Disktools      Help          News          demo.txt
Documents      Motif         UsrAdmin       demo.tex
zeus:/home/uhl>
```

Most commands also allow parameter transfer. Thus data and options can be transferred to a program that can evaluate them.

options

```
zeus:/home/uhl>ls -l
total 1516
drwxr-xr-x      3 uhl      users      1024 May 14  1994
Disktools
drwxr-xr-x      2 uhl      users      1024 May  1  1994
Documents
drwxr-xr-x      2 uhl      users      1024 Aug  8 12:28
Help
```

```
drwxr-xr-x      2 uhl      users      1024 Dec 24 17:27
Motif
drwxr-xr-x      2 uhl      users      1024 Jul 31 12:56
News
drwxr-xr-x      3 uhl      users      1024 Mar 27 1994
UsrAdmin
-rw-r--r--      1 uhl      users      1474560 Dec 29 21:10
demo.tex
-rw-r--r--      1 uhl      users      1474560 Dec 29 21:10
demo.txt
zeus:/home/uhl>
```

ls, -l With the `ls` command, the parameter `-l` leads to a more comprehensive version of the table of contents.

Environment

environment Another way to transfer data to program is through the environment. Here a list of variables and their respective values is automatically passed to a program every time it is launched. In order to obtain a list of the currently defined environment variables, use the command `set` in the Bourne shell:

```
zeus:/home/uhl> set
PS1=$HOST:$PWD>
PS2->
PATH
PATH=/bin:/usr/bin:/usr/local/bin:
PWD=/home/uhl
TERM=vt100
UID=401
zeus:/home/uhl>
```

variables These variables can be deleted and redefined as needed. For example, you can redefine a variable called `AUTO` as `VW` and pass it to another program as follows:

```
zeus:/home/uhl> export AUTO=VW
zeus:/home/uhl> set
PS1=$HOST:$PWD>
PS2->
PATH=/bin:/usr/bin:/usr/local/bin:
PWD=/home/uhl
TERM=vt100
UID=401
AUTO=VW
zeus:/home/uhl>
```

PATH, PS1 Environment variables are generally used to specify global system settings such as the access path for commands (`PATH`) or the nature of

the prompt (`PS1`). These settings can be retained in a special file called `.profile` in the user's home directory to avoid repeated manual entry. This file is automatically executed at every login.

prompt
`.profile`

```
#
# Beispiel einer .profile-Datei
#
PATH=$PATH:/usr/local/bin
AUTO=vw
PS1='$HOST:$PWD>'
```

The Bourne shell also recognizes another start-up file called `.bashrc`, which is activated not only at login but also every time a new shell is started. To be sure that certain environment variables are definitively set, define them in the `.bashrc` file.

`.bashrc`

Redirection

The concept of standard input (`stdin`) and standard output (`stdout`) is basic for UNIX. This normally involves the keyboard and the console. Simple commands such as `ls` produce their results as standard output. A shell allows for the redirection of standard input or standard output in a file without involving the command. To do this, you use the operators `>` and `<`:

`stdin`, `stdout`

redirection

```
zeus:/home/uhl> ls > list.txt
zeus:/home/uhl>
```

As the above example shows, there is no console output in this case. The output is redirected to a file called `list.txt`, which can be displayed using `cat`:

file

```
zeus:/home/uhl> cat list.txt
Disktools
Documents
Help
Motif
News
UsrAdmin
demo.tex
demo.txt
zeus:/home/uhl>
```

Commands that expect data from standard input (e.g., `cat`) can also receive redirected input from a file:


```
zeus:/home/uhl> cat < liste.txt
Disktools
Documents
Help
Motif
News
UsrAdmin
demo.tex
demo.txt
zeus:/home/uhl>
```

Aside from standard input and standard output, there is also a third channel that is linked to the console. Since it is used for the output of error messages, it is called standard error channel (`stderr`). Even if the user redirects `stdin` and `stdout`, error messages will still be output to the console. The individual channels can be addressed through their file descriptors.

Term	Abbreviation	File descriptor	Standard device
Standard input	<code>stdin</code>	0	keyboard
Standard output	<code>stdout</code>	1	console
Standard error	<code>stderr</code>	2	console

For instance, the following instruction is enough to redirect `stderr` to a file:

```
zeus:/home/uhl> cat liste.txt 2>error.txt
```

Of course, `stdout` and `stderr` can both be redirected to a file:

```
zeus:/home/uhl> cat 2>&1 >output.txt
```

Here `stderr` is first redirected after `stdout`; then `stdout` is redirected to a file.

Often the existing contents of a file need to be retained when `stdout` is redirected. To achieve this, use the special operator `>>`, which appends the redirected output to an existing file rather than overwriting the file.

The possibility of using the standard output of one command as input for another one is even more interesting. This can be done with the *pipe* operator | :

pipe

```
zeus:/home/uhl> ls | wc
      8      8      63
zeus:/home/uhl>
```

Here the command `wc` counts the number of words, lines, and characters that it has received from the standard input, thus ascertaining the number of files in the current directory. The possibility of using links to form new commands out of a series of short commands makes the shell a very powerful tool.

linked commands

File Name Expansion

You may often find it more practical to transfer several files to a single command at once rather than entering each file name individually. This can be done using wildcards, which have a kind of joker function. If a transfer parameter contains a wildcard, the shell checks which file names in the appropriate directory match the search pattern and transfers these to the command:

wildcard

```
zeus:/home/uhl> cat *.txt
```

In this example, all the files ending with `.txt` are transferred to the command `cat`. The most important wildcards are the following:

Character	Function
*	replaces any characters (or none)
?	replaces any one character
abc...	replaces one character from a defined quantity
[a-z]	areas can be defined with a hyphen
[!abc...]	replaces all characters not contained in the defined quantity

Quoting

metacharacters

In addition to wildcards, the shell also recognizes a number of other metacharacters. These need to be *quoted*, however, to cancel the special meaning of these characters. You can do this by placing a backslash in front of the character:

```
zeus:/home/uhl> echo \?\?\?  
???  
zeus:/home/uhl>
```

single / double quotes

To transfer a series of individual parameters to a command as a character string, enclose them in single quotes or double quotes:

```
zeus:/home/uhl> echo "Hello World!"  
Hello World!  
zeus:/home/uhl> echo 'Hey "you" there!'  
Hallo "Du" da!  
zeus:/home/uhl>
```

As this example demonstrates, single quotes cancel the effect of double quotes.

backquoting

Backquoting is the third form of quoting. This refers to the possibility of treating the output of a command like a character string.

```
zeus:/home/uhl> cp 'which ls' .
```

The above command copies the program `ls` in the current directory. `which` provides the access path for `ls`, which is used as the first parameter of the copy command. Alternatively, a variable could have been assigned to it first:

```
zeus:/home/uhl> path='which ls'  
zeus:/home/uhl> echo $path  
/bin/ls  
zeus:/home/uhl> cp $path .
```

Short Commands

alias

Aliases can abbreviate frequently used commands. Create a new alias with the command of the same name:

```

zeus:/home/uhl> alias l='ls -l'
zeus:/home/uhl> l
total 1516
drwxr-xr-x      3 uhl      users      1024 May 14 1994
Disktools
drwxr-xr-x      2 uhl      users      1024 May  1 1994
Documents
drwxr-xr-x      2 uhl      users      1024 Aug  8 12:28
Help
drwxr-xr-x      2 uhl      users      1024 Dec 24 17:27
Motif
drwxr-xr-x      2 uhl      users      1024 Jul 31 12:56
News
drwxr-xr-x      3 uhl      users      1024 Mar 27 1994
UsrAdmin
-rw-r--r--      1 uhl      users      1474560 Dec 29 21:10
demo.tex
-rw-r--r--      1 uhl      users      1474560 Dec 29 21:10
demo.txt
zeus:/home/uhl>

```

In the Bourne shell you can get a list of the currently defined aliases by entering `alias` without parameters:

```

zeus:/home/uhl> alias
alias l='ls -l'
alias ll='ls -laF'
zeus:/home/uhl>

```

To avoid having repeatedly to enter the same relatively long command lines, create a shell script. A shell script can be executed like a normal UNIX command. Produce the shell script by creating a file with the desired name and containing the individual instructions:

shell script

```

#!/bin/sh
#
# filecount: Zeigt die Anzahl der Dateien im aktuellen
#            Verzeichnis
ls | wc

```

The first line (`/bin/sh`) makes the above script run in a Bourne shell. The file's permissions have to be changed in order to allow execution of a shell script:

```

zeus:/home/uhl> chmod +x filecount
zeus:/home/uhl> filecount
      8      8      63
zeus:/home/uhl>

```

programming
booting

However, it is also possible to have instructions in a shell script that are considerably more complex than just calling up a few commands. The Bourne shell's script language also allows the construction of loops, tests, branches, and expressions. This means that you can program fairly complicated routines. Large parts of the UNIX system were developed on this basis. Particularly when the system is booted, it goes through a number of different shell scripts. The following provides a general overview of the development of such scripts.

Variables

local variables
export

Aside from the environment variables that are passed on to a program that is invoked, as mentioned above, there are also local variables within the shell. You can turn local variables into globally recognized environment variables by placing the key word `export` in front of them. Variables can accommodate any character string:

```
COMPUTER=IBM
```

access to variables

In this example, the value `IBM` is assigned to the variable `COMPUTER`. Then if access to the contents of an (environment) variable is needed, a `$` sign has to be put in front of the name:

```
echo $COMPUTER
```

command line

A series of special variables are already predefined within a shell script. These may contain, for example, the parameters from the command line:

\$#	number of transferred parameters
\$0	name of the shell script
\$n	nth parameter
\$*	all parameters
\$\$	ID of the current process
\$?	return value of the command last executed

If the value of a variable within double quotes (") is accessed, then this value is replaced as usual by the current value. You can avoid this by using single quotes:

double quotes

```
zeus:/home/uhl> echo "Terminal: $TERM"
Terminal: xterm
zeus:/home/uhl> echo 'Terminal: $TERM'
Terminal: $TERM
zeus:/home/uhl>
```

Keyboard input

The Bourne shell provides the instruction `read` for interactive data entry by the user. As a parameter, `read` expects the name of a shell variable under which the user's input is to be filed.

`read`

```
echo -n "Input: "
read line
echo $line
```

The program fragment above outputs a prompt, reads a line from the standard input, and files this under the variable `line`, which is output in the last command line.

prompt

Branches

The Bourne shell provides IF for creating simple branches. Thus a routine can be made dependent on certain conditions. If the first condition is met, then only `commands1` executes. On `condition2`, `commands2` executes. `commands3` executes if neither of the conditions holds.

IF branch

```
if condition1
then
    commands1
[ elif condition2
then
    commands2 ]
...
[ else
    commands3 ]
fi
```

In general, a condition consists of calling up an external program. If this returns zero, then the condition has been met. `test` is a special

`test`

command for this purpose; it can also be accessed with `[`. A simple comparison of two character strings then looks like this:

```
if [ "$car" = "vw" ]
then
    echo "You have bought the right car!"
else
    echo "Buy a different car!"
fi
```

The `test` command also recognizes quite a few other arguments, which can be found in the reference section on page 402. The `case` instruction provides another form of branching. Regular expressions (see section 2.8) are used here to differentiate between the individual variations in a routine:

```
case Value in
    Expression1)
        Commands1 ;;
    Expression2)
        Commands2 ;;
    ...
    *)
        Commands3 ;;
esac
```

OR It is also possible to link several regular expressions with the `|` sign (OR):

```
while true
do
    echo -n "* "
    read line
    case "$line" in
        monitor|bildschirm)
            echo screen ;;
        auto)
            echo car ;;
        haus)
            echo house ;;
        ENDE)
            exit 0 ;;
        *)
            echo "Word unknown!" ;;
    esac
done
```

Loops

FOR loop FOR loops provide the possibility of executing the same string of instructions several times for each parameter of the command line or for a password list.

The general syntax of a FOR loop looks like this:

```
for x [in list]
do
    Instructions
done
```

If no list is named, then the instructions are executed once for each parameter of the command line:

command line

```
for i
do
    echo $i
done
```

This outputs the parameters of the command line successively.

```
for i in audi bwm mercedes volvo vw
do
    echo $i
done
```

It is somewhat more practical, though, to use a script that adds the extension `.txt` to all the files ending with `.doc`:

extension

```
for i in *.doc
do
    echo $i
    tmp='basename $i .doc'
    mv $i $tmp.txt
done
```

WHILE loops keep repeating a block of instructions until the passed condition is no longer met. Like the IF instruction, this type of condition is an external command:

WHILE loops

```
while condition
do
    commands
done
```

It could be used, for example, as follows:


```
while [ "$line" != "ENDE" ]
do
    echo -n "*"
    read line
    echo $line
done
```

The line that is read from the standard input is output repeatedly until the user enters `DONE`.

2.8 Search patterns

metacharacters

Search patterns can be specified not only in the shells but also in the editors Emacs and vi and in search programs like grep. This is done using metacharacters, primarily `?` and `*`. The meaning of these characters is different, however, for the shells and for other programs.

regular expressions

We need to differentiate between simple wildcards, as they are used to denote file names, and regular expressions. While the latter are more complicated, they do allow for the specification of more complex search patterns. The procedure for replacing these metacharacters is usually called *globbing* with shells and *pattern matching* with regular expressions.

globbing

We have already discussed how to work with wildcards in the shells on page 25, so we will not go into that again here. At the moment, regular expressions are of greater interest. The following list provides an overview of the meanings of the metacharacters:

- . Any given character.
- * Any given number of the times that a preceding character or expression occurs. For example, `a*` would apply to any length of a series of the letter `a`, including no `a`.
- + The preceding character or expression must occur at least once.
`a+` means one or more `a` characters.
- ? The last character or expression occurs exactly once or not at all.
- ^ Beginning of the line. This is used to create expressions that may occur only at the beginning of a line. For example, `^ab*` represents an `a` at the beginning of a line, which may be followed by any number of occurrences of the letter `b`.
- \$ End of the line.

- [] Any of the characters inside the brackets. The characters ^ and - play a special role here. If the contents of the brackets begins with ^, then the statement is negated. All characters are permitted except for those stated. A range of characters can be defined with the character -. [A-Z] stands for any capital letters, and [^a-c] stands for any characters except a, b, and c. [123] stands for any of the characters 1, 2, or 3.
- \— Cancels the special meaning of the subsequent character. This makes it possible to use metacharacters themselves in expressions. ???*+ means that * occurs at least once.
- () Encloses a regular expression. This means, for example, that the characters * and + may also be used for other expressions.

The meaning of these characters may vary slightly depending on the program, and additional metacharacters may be defined as well. When in doubt, check the manual. The following examples illustrate the use of regular expressions with the example of the command `grep`. This may need to be invoked with the option `-E`, allowing enhanced expressions. The relevant part of the output is printed here in bold type for easier understanding.

variations

grep

```
hermes:/home/strobel> grep The test file
this is described in the manual.
You may need to call it up with the option -E

hermes:/home/strobel> grep ^Die testfile
Dieser muß eventuell mit der Option -E aufgerufen werden,

hermes:/home/strobel> grep ren testfile
kann die Bedeutung dieser Zeichen leicht variieren
regulären Ausdrücke am Beispiel des Befehls grep.

hermes:/home/strobel> grep 'ren$' testfile
kann die Bedeutung dieser Zeichen leicht variieren

hermes:/home/strobel> grep -E "1+ ?m" testfile
Dieser muß eventuell mit der Option -E aufgerufen werden,

hermes:/home/strobel> grep -E '(ll)+[^\$]' testfile
Dieser muß eventuell mit der Option -E aufgerufen werden,

hermes:/home/strobel>
hermes:/home/strobel> grep 'B.*1' testfile
Je nach Programm kann die Bedeutung dieser Zeichen leicht
folgenden Beispiele verdeutlichen die Verwendung der
regulären Ausdrücke am Beispiel des Befehls grep.
```

The search pattern `1+ ?m` stands for one or more occurrences of the letter 1, followed by an optional space and the letter m. `(ll)+[^\$]`

means that the expression `ll` occurs at least once, but it must not necessarily be followed by the end of the line. This expression must be enclosed in simple quotes, so that the shell does not attempt to process the expression, but rather passes it to the command `grep` directly.

2.9 Daemons

background Daemons are special processes that run in the background and usually assume important tasks in a UNIX system. With daemons, large parts of the operating system run as independent programs. This keeps the operating system kernel relatively small. Furthermore, **activate** individual daemons can be activated, or restarted after a change in configuration, even during operation. Since daemons run as independent processes, they can run parallel to one another; thus, do not block other programs. The following sections illustrate some examples of daemons.

Printer daemon (lpd)

printer daemon At regular intervals the line printer daemon (`lpd`) checks the directory `/usr/spool` for new printing jobs and sends any such jobs to the respective printer. For outputting a file on a printer, Linux **lpr** provides the `lpr` command known from BSD UNIX. New print jobs are normally appended to the end of the queue before they are sent by the printer daemon.

Cron daemon

If a user wants to execute a program at certain times or at regular intervals, the cron daemon makes this possible. For each user, this daemon manages a table in which the times are entered when the desired processes are to start. The output of an executed command or the respective error messages are sent to the user as e-mail. If a script **e-mail** is to be executed only once at a certain time, the command `at` does the job. Repeated execution of a process at regular intervals requires **at** an entry in the user's cron daemon table (`crontab`). A separate **crontab** command named `crontab` serves this purpose.

Syslog daemon

Since a daemon normally does not send output to the console, a separate protocol daemon was created to handle output and error messages from other daemons. This output can be displayed on the console, written to a file, or forwarded as e-mail to the system administrator.

protocol

2.10 Overview of commands

To help newcomers become familiar with Linux, we provide a collection of the most important UNIX commands, along with brief explanations. More detailed information is available in the standard UNIX literature or the on-line manual.

- **ls** - outputs a list of files and directories. As an option, the file sizes, corresponding permissions, and file owners can be displayed. Recursive output of complete directory trees is also possible.
- **cd** - If no parameter is specified, the current directory becomes the home directory
- **cp** - copies the specified files from one directory to another directory or to another file. As an option, a complete directory tree can be copied recursively.
- **mv** - moves a file within a file system, or renames a file or a directory.
- **rm** - removes a file. As an option, an entire file tree can be deleted recursively.
- **mkdir** - creates a new directory.
- **rmdir** - removes an empty directory.
- **exit** - exits the current shell.
- **more** - displays the contents of a text file page by page on the console. In addition, character strings can be searched for in the file.
- **man** - displays the on-line documentation (Manual pages corresponding to a command).
- **cat** - intended for the concatenation (appending) of text files, but can also be used to output a file.

- **grep** – searches within the specified files for a certain pattern.
- **passwd** – changes a user's password.
- **ps** – lists all running processes with their process ID.
- **kill** – terminates the process with the specified process ID.
- **su** – temporarily changes the user ID, without having to repeat the login. If `-` is specified as an additional parameter, a renewed login is triggered.

Linux Features

This chapter assumes that the reader already has a basic knowledge of UNIX or has read the previous chapters. Here we describe in more detail some of the important characteristics and features of Linux that distinguish this system from other UNIX variants and from other PC operating systems.

3.1 Virtual consoles

Many PC UNIX implementations support *virtual consoles*, which provide the capability to manage multiple independent login sessions on one console. Switching between the individual sessions usually occurs via a special key combination.

multiple logins

Under Linux the **<Alt>** key combined with a function key enables switching between virtual consoles. The maximum number of virtual consoles is determined in the kernel. The file `/etc/inittab` establishes the configuration for which of these consoles is to display the login prompt.

kernel

Under the X11 System the **<Alt>** key is reserved for applications. The X Window System handles the switching to another virtual console with the three-way combination of **<Ctrl-Alt>** and the respective function key corresponding to the virtual console number. This allows the user to switch between the graphical interface of X Window System and the text-oriented interface of the virtual consoles.

X11

function keys

It is even possible to start multiple X servers. However, this is not recommended because there is seldom enough memory and thus performance suffers noticeably. Instead, use a virtual window

manager under X Window System, such as `olvwm` or `fvwm`, which also offers multiple virtual consoles.

3.2 Linux file systems

file systems The multitude of available file systems under Linux might seem confusing at first glance. The following subsections list these file systems and describe their most important features.

MINIX file system

first file system The first Linux versions furnished only one type of file system, and it relied heavily on the MINIX file system. This circumvented the effort required for a completely new development. Furthermore, this provided a stable file system from the start, although the MINIX file system certainly has some significant drawbacks as well.

14 characters File names cannot exceed 14 characters in length and the size of
64 MB partitions a partition is restricted to 64 MB. Although newer versions of this Linux/MINIX file system permit longer file names (30 characters), this file system is meanwhile scarcely in use.

symbolic links However, it is noteworthy that, in contrast to many proprietary System V implementations, even this first version of a Linux file system also supported symbolic links.

Extended file system (ext)

Remy Card To overcome the above restrictions, a Frenchman named Remy Card implemented the first alternative file system. His Extended File System (`ext`) for the first time supported files and partitions of up to 2 GB. He also raised the maximum length of file names to 255 characters.

i-nodes But this system also has its weaknesses. Free blocks and i-nodes are not managed in a bit vector, but in a linked list.
fragmentation After a longer period of operation, this leads to extensive fragmentation of the memory, which causes noticeably longer access times.

Extended2 file system (ext2)

From the Extended File System, the Extended2 file system evolved after some time; this is currently the most widespread file system

under Linux. The fragmentation problems no longer occur in this system, and the limitation of 2 GB for file systems was canceled. Furthermore, Extended2 supports a mechanism that saves lost sectors in a special directory (`lost+found`). A possible system crash and its resulting corrupt file system are recognized when the system is started, and the damage can be repaired with a special utility (`e2fsck`).

less fragmentation

lost + found

Xia file system

The Extended2 file system has not been the only attempt to establish a new, faster file system. At nearly the same time the Xia file system, named after its author, Frank Xia, appeared. This file system also increased the maximum partition size to 4 GB. File names can be up to 248 characters long. The size of a file is currently limited to 64 MB at this time.

Frank Xia

64 MB file size

Other file systems

The DOS file system permits Linux transparent access to DOS diskettes or partitions (and OS/2 FAT partitions). This permits access to pre-existing data, as explained in section 3.3.

DOS & OS/2

To access Xenix, System V and OS/2-HPFS partitions, special file systems were developed, but some of these are not yet fully implemented.

System V
OS/2

ISO 9660/HighSierra file system

To provide access to CD-ROMs, Linux provides both ISO 9660 and High Sierra file systems. Likewise the Rockridge Extensions supporting longer file names have been implemented.

CD-ROM
Rockridge Extensions

Proc file system

This process file system does not manage physical files, but enables access to data in the kernel and the currently running processes. On system startup the proc file system is normally written to the subdirectory `/proc` in the root directory. For every running process, this directory contains a subdirectory whose name is the respective process ID. The files that it contains provide a flexible interface to the actual process-specific information. In general these are virtual

kernel data
process ID

ASCII

ASCII files whose contents can be output with the `cat` command. This allows the user to determine the contents of the command line or the environment variables of a process. Information about memory requirements, the parent process or the current process state can be extracted in this way.

3.3 Data exchange

DOS, Windows Linux is seldom the sole operating system on a PC. More frequently another partition or hard disk contains DOS with MS-Windows, OS/2 or another PC UNIX variant. A user switching from DOS to Linux is seldom willing to sacrifice the old programs.

The following subsections show how Linux is able to work with other operating systems and exchange data and programs with them.

boot manager When using operating systems on one computer, a boot manager proves essential. At system startup, this utility enables the user
LILO to select the operating system to be booted. The Linux Loader (LILO) fulfills this function, among others. Chapter 5, "Installation," describes the installation and operation of LILO.

MTools

DOS Since particularly the exchange of files with an DOS system is required nowadays of nearly all operating systems, for some time there have been freely available programs for processing DOS files under UNIX. As with many other UNIX systems, Linux provides the
mdir, mcopy MTools for this purpose. These are commands like `mdir` and `mcopy` that support the reading of the directory of an DOS storage medium (typically a diskette) and the copying of files. The following is an example of accessing a DOS diskette with MTools:

```
dirkl:/home/stefan# mdir a:
Volume in drive A is dosdisk1
Directory for A:/

COMMAND  COM      55591    3-10-93    6:00a
WINA20    386      9349     6-11-91   12:00p
AUTOEXEC BAT      359     8-26-93    9:02p
CONFIG    SYS      377     5-23-93    2:48p
DOSKEY    COM     6012     6-11-91   12:00p
EDIT      COM      429     6-11-91   12:00p
FORMAT    COM    34223     6-11-91   12:00p
          9 file(s)    1270784 bytes free
dirkl:/home/stefan# mcopy -t a:autoexec.bat .
```

```

Copying AUTOEXEC.BAT
dirk1:/home/stefan# mdel a:autoexec.bat
dirk1:/home/stefan#

```

The disk drive device needs to be free, so that you can access the Mtools on a diskette. That means that the diskette must not be mounted.

DOS file system

In addition to MTools, Linux provides another method to access DOS storage media, the DOS file system. This makes it possible to mount diskettes and DOS partitions of a hard disk in the same way as other file systems in the Linux directory tree. This provides completely transparent access to the contained files. Access is faster than with MTools because the input and output operation can now profit from the cache of the operating system. The following is an example of accessing an DOS partition with the DOS file system:

diskettes
DOS partitions
speed
cache

```

dirk1:/# cd msdos
dirk1:/msdos# ls -a
./
dirk1:/msdos# cd ..
dirk1:/# mount -t msdos /dev/hda2 /msdos
dirk1:/# cd msdos
dirk1:/msdos# ls -a
./          command.com*  format.com*  tools/
../         config.sys*   io.sys*      wina20.386*
autoexec.bat* dos/         msdos.sys*   windows/
dirk1:/msdos# cd dos
dirk1:/msdos/dos#

```

However, the drawback of mounting diskettes is that they can no longer be inserted and removed at will; instead, the commands `mount` and `umount` must be invoked with each change of diskettes. If diskette is removed while it is still mounted, then the diskette inserted subsequently is usually overwritten.

mount and umount

Since DOS provides neither user IDs nor group IDs, the individual files of a mounted DOS file system cannot be assigned individual settings for access control. However, on mounting the volume, group and user IDs as well as access privileges for the overall file system can be specified as an option. This at least permits control of access to the file system as a whole.

access privileges
user ID

section

A complete description of how the DOS file system functions under Linux and the available options can be found in the on-line Manual page for `mount` and the file `README.dosfs`, which resides on ftp servers along with the source code of the file system. There are two on-line Manual pages for `mount`, one for the command and one for the routine in the C library. To display the correct on-line Manual page, the section must be specified in the invocation of the `man` command. The invocation for the `man` command would be:

```
man 8 mount
```

The most important options that can be specified in the invocation of the `mount` command for a DOS file system are:

- `uid=<number>`
- `gid=<number>`
- `umask=<number>`
- `conv=binary or text or auto` (see next paragraph)

The following example shows the mounting of a DOS file system with the specification of options (described on page 390 in the Reference section):

```
stef1:/# mount -t msdos -o umask=000 /dev/hda1 /dos
```

The options can also be specified in the file `/etc/fstab`:

<code>/dev/hda3</code>	<code>none</code>	<code>swap</code>	<code>defaults</code>
<code>/dev/hda2</code>	<code>/</code>	<code>ext2</code>	<code>defaults</code>
<code>/dev/hda1</code>	<code>/dos</code>	<code>msdos</code>	<code>rw,umask=000</code>
<code>none</code>	<code>/proc</code>	<code>proc</code>	<code>defaults</code>

Text conversion

A fundamental problem in data exchange between DOS and UNIX is their different representations of new lines in text files with respect to carriage return (CR) and line feed (LF). In `mcopy` this has been

new line (CR/LF)

solved with the command line option `-t`, which specifies whether to copy a file in binary mode or in text mode with conversion.

conversion

Since a mounted DOS file system allows access to many different files, there is no universal solution. Although there are options for the `mount` command that activate automatic conversion, this process does not always function reliably. Whether a file is a text file or a binary file cannot always be decided with certainty.

options

3.4 Loadable Modules

Classical UNIX systems generally possess a monolithic kernel. The entire kernel needs to be newly linked if a new driver is to be integrated into the system. Although the architecture of the Linux kernel is very similar in principle, it does provide *loadable modules*. This term refers to object files that can be loaded or removed while the system is running. Numerous Linux drivers are already available as loadable modules. The iBCS2 emulator (see section 4.3) is also integrated this way.

monolithic kernels

loading while running

The following commands are used for the administration of modules:

<code>insmod <module></code>	integrates the passed module into the system
<code>depmod</code>	erstellt eine Liste von Abhängigkeiten zwischen den einzelnen Modulen
<code>modprobe <module></code>	lädt das übergebene und alle von diesem benötigten Module
<code>rmmod <module></code>	removes the passed module
<code>lsmod</code>	outputs a list of currently loaded modules

Normally, the modules that will be needed are loaded in one of the `rc` files (see section 7.2) when the system is started. In general, the command `modprobe` should be used to load a module because this command automatically loads all additional modules that the module requires.

3.5 Sound

PC sound boards
kernel

Unlike a number of other PC-based UNIX systems, Linux supports all the common PC sound boards. The necessary drivers are available in the kernel code; however, they need to be configured (`make config`) before compilation. As soon as the appropriate driver is present in the kernel, a series of new devices become available:

<code>/dev/mixer</code>	mixer for various audio channels
<code>/dev/audio</code>	Sun audio device (μ -law Format)
<code>/dev/sequenzer</code>	sequencer device
<code>/dev/midi</code>	device for playing MIDI data
<code>/dev/sndstat</code>	status device for audio system
<code>/dev/dsp</code>	audio device (raw-Format)

playing and recording

A series of programs exist for recording and playing audio data (`vrec` and `vplay`). In the simplest cases, however, the `cat` command is enough to record

```
zeus:/home/uhl> cat < /dev/audio > sound.au
```

or to play audio files:

```
zeus:/home/uhl> cat sound.au > /dev/audio
```

3.6 Alternative shells

csh

Because their source codes are not freely available and because appreciably more comfortable alternative shells have been developed, Linux seldom uses the original UNIX shells `sh` and `csh`. The following are examples of such alternative shells.

bash

GNU

The `bash` (Bourne Again shell) evolved, like many other programs, from the GNU project of the Free Software Foundation. It can be considered an extension of the Korn shell, which shares many of the

features of the `tcsh` (see the next section). Furthermore, the scroll keys allow activation of practical special functions like automatic name and path extension and history scrolling. The following example explains this feature in more detail.

To change from the directory `/home/stefan` to the directory `/usr/src/linux`, the command `cd /usr/src/linux` could naturally be entered manually, letter by letter. With automatic path recognition in `bash` it suffices to specify directories only enough to assure that they are unambiguous. Entering **<Tab>** then automatically extends the path.

`tcsh`

automatic path
extension

```
/home/stefan> cd /usr/s
```

After entry of **<Tab>** the above line becomes:

```
/home/stefan> cd /usr/src/
```

Then an `l` (`el`) is entered:

```
/home/stefan> cd /usr/src/l
```

After entry of **<Tab>** the above line becomes:

```
/home/stefan> cd /usr/src/linux/
```

In order to edit the makefile of the Linux kernel in this directory, it suffices to enter only a few characters:

```
/usr/src/linux> emacs M
```

After entry of **<Tab>** the above line becomes:

```
/usr/src/linux> emacs Makefile
```

Makefile is the only file in this directory that begins with a capital M.

If an attempt to make the extension results in multiple alternatives, there is a warning sound, and, after the **<Tab>** key is pressed again, `bash` displays all possible variants.

variants
command history

The up and down scroll keys allow the user to move through the command history. In the above example, pressing the scroll up key once would yield the following on the command line:

```
/usr/src/linux> emacs Makefile
```

Pressing the scroll up again would display the following:

```
/usr/src/linux> cd /usr/src/linux/
```

environment variables

Environment variables enable the display of the host name and the current path in the prompt. To accomplish this, the following line is added to the file `.bashrc` in the respective home directory:

```
PS1='""h:$PWD>'
```

Bourne Again shell

The Bourne Again shell affords many other features that cannot be described here in detail. Please refer to the on-line Manual pages for more information on `bash`.

tcsh

One alternative to the Bourne Again shell is `tcsh`, an extension of the C shell. Like `bash` it can automatically extend paths and commands when the **<TAB>** key is pressed, and the scroll keys allow moving through the command history and editing the command line. Furthermore, various additional functions can be activated, such as watch mode, where a message is output if a user logs in or out. The display of all alternatives for automatic extension is invoked in `tcsh` with **<Ctrl-D>**, but all keys can be configured. The directory listing of the directory `/usr/src` can be displayed without fully entering the command `cd`:

C shell
command history

automatic extension

```
linux1:/home/tul>cd /usr/src/ <Ctrl D>
```

An interesting feature is the automatic correction of erroneously entered commands or file names. Here `tcsh` attempts to use the next closest command or file name. This feature is activated by entering **<Meta-s>**. correction

3.7 Extended commands

Many standard UNIX commands that Linux uses emerged from the GNU project and are extensions of the normal commands. For example, under Linux the command `ls` has more than 20 options with which the kind of display, sorting, or handling of symbolic notes can be modified as needed. Many distributions even contain color variants of the `ls` command that displays the lines of the directory listing by color according to the type of the directory element (link, executable file, tar file, subdirectory, etc.). GNU options

Here is an excerpt from the on-line Manual page of the GNU `ls` command:

```
LS(1L) LS(1L)
NAME
    ls, dir, vdir, ll, lsf - list contents of directories
SYNOPSIS
    ls [-abcdgiklmnpqrstuxABCFILNQRSUX1] [-w cols] [-T cols]
    [-I pattern] [--all] [--escape] [--directory] [--inode]
    [--kilobytes] [--numeric uid:gid] [--hide-control-chars]
    [--reverse] [--size] [--width=cols] [--tabsize=cols]
    [--almost-all] [--ignore-backups] [--classify] [--file-
    type] [--ignore-pattern] [--dereference] [--literal]
    [--quote-name] [--recursive]
    [--sort={none,time,size,extension}] [--for-
    mat={long,verbose,commas,across,vertical,single-column}]
    [--time={atime,access,use,ctime,status}] [path...]
```

GNU tar

The GNU variant of the `tar` command has the option `z` for automatic compression and decompression of tar archives, and the option `M` for establishing a multivolume archive (a single archive that is distributed on multiple storage media). compression archive

gzip

compress The command `gzip` replaces `compress` under Linux. This program is compatible to many other compression techniques and has a

efficiency significantly higher efficiency than `compress`. The command `tar` enables the combination of individual files and subdirectories into a single archive. A compressed tar archive that was compressed with the UNIX command `compress` had a size of 1.5 MB; `gzip` frequently requires only 900 KB.

source codes All these extended commands are naturally available in source code and can be compiled on other UNIX machines. The advantage

Linux of Linux is that these are used from the start and no additional effort has to be invested in the configuration, compilation and installation of these utilities.

reference The list of all enhanced commands and their options would certainly fill a book itself. The most important ones are listed in the Reference Section. Refer to the on-line Manual pages and the individual commands there for more complete details.

Emulators

The various emulators available to Linux users are increasing in importance. In the first edition of this book we described the DOS emulator within the chapter on Linux features. Meanwhile various other emulators have become available. Many such projects are no longer limited to Linux, but are being developed in parallel for other commercial and freeware UNIX variants such as Free BSD.

4.1 DOS emulator

As for OS/2 and other newer operating systems, Linux has a DOS emulator that permits running DOS programs simultaneously with other Linux applications. However, this emulator does not emulate the DOS operating system, but only the rudimentary input/output routines (BIOS); it enables access to all important devices. With this system extension, DOS can be booted under the control of Linux. The particular DOS version is of subordinate importance.

DOS programs

BIOS
booting DOS

Overview

Although the emulator is still far from being able to run every DOS program without problems, the possibilities are quite impressive. Programs can run in graphic mode and with the support of high memory (HMA), upper memory blocks (UMB) and expanded memory (EMS). Support of the DOS protected mode interface (DPMI) and access to Novell servers within the emulator are also possible, but are still under development. DOS programs such as Norton Commander and the text editor QEdit work without problems, and even larger commercial products such as Turbo Pascal and Word Perfect can be used.

graphics
HMA, UMB
EMS
DPMI



Figure 4.1. DOS emulator under X11

X11 console The emulator can run either under X11 as a separate window or in a virtual console in text mode. You can switch consoles within the emulator; as under X11, you simultaneously press `<Ctrl>` and `<Alt>` along with a function key.

image files In addition to the normal diskettes and DOS partitions, the DOS emulator can also access the Linux file system and disk image files. DOS programs treat these files like real data media. The are used primarily for booting the DOS emulator.

To configure the DOS emulator, you edit the file `/etc/dosemu.conf`. You also need a file named `/etc/dosemu.users`, which contains the names of all users with permission to use the emulator.

Disk image files are normally stored in the directory `/usr/lib/dosemu` or `/var/lib/dosemu`.

hardware BIOS The DOS emulator, as mentioned above, does not emulate the whole DOS operating system, but only the hardware and the BIOS of a PC. Thus, starting the emulator requires a DOS boot diskette, a hard disk partition with DOS installed, or a corresponding image file.

Direct access to a DOS partition of a hard disk and booting from this partition is one alternative. In this case, however, the partition must not be mounted in the Linux file system, since this can cause conflicts that can mean data loss.

conflicts booting from an image file The alternative is booting from a diskette or a special image file, which the DOS emulator uses like a real diskette or hard disk.

Naturally, using a file in place of a boot disk affords an elegant and fast solution, but such a file must be created first.

Creating an image file for booting

Creating the boot image file for booting requires a normal bootable DOS diskette. Several special DOS programs that accompany the emulator also need to be copied onto this diskette. These programs allow access to the Linux file system from within the emulator and permit reading or modifying certain settings of the emulator. We also recommend copying a simple editor onto the diskette.

DOS diskette
programs

editor

The following terminal capture shows the subdirectories of the DOS emulator distribution containing the driver and programs:

```
stef1:strobels/dosemu0.50pl1/commands> ls
Makefile      dosdbg.c      exitemu.S      lredir.exe
vgaon.S       bootoff.S     dosdbg.exe     exitemu.com    lredir.readme
vgaon.com     bootoff.com   dosdbg.readme  lancheck.exe   pdipx.com
booton.S      dumpconf.asm  linpkt/        vgaoff.S
booton.com    dumpconf.exe  lredir.c       vgaoff.com
stef1:strobels/dosemu0.50pl1/commands>
```

If these files are not included in the distribution, then the complete emulator distribution can be copied from one of the usual FTP servers. On the server `sunsite.unc.edu`, for example, the DOS emulator distribution resides in the directory `/pub/Linux/system/emulators/dosemu`. To copy the files from Linux to a DOS diskette, use MTools or mount the diskette (see section 3.3).

FTP server

MTools

After all programs and files have been copied to the boot diskette, it can be copied to a file using the command `dd`, as in the following terminal capture. This file can be stored in any directory (bdisk in this example). If it is for personal use, employ the home directory; if multiple users will be using the DOS boot image, a system subdirectory should be created, such as `/var/lib/dosemu`.

dd

```
zeus:/home/strobels> dd if=/dev/fd0 of=bdisk bs=16k
```

Now the configuration file for the DOS emulator must be modified so that this image file can be used for booting. The corresponding entries in the configuration file are:

```
boota
bootdisk {heads 2 sectors 18 tracks 80 threeinch file bdisk}
```

Unless the image file is located in the current directory, its complete path must be specified in the configuration file.

Theoretically, you could start the DOS emulator now by entering `dos`. For practical use, however, you still need to make some settings.

Accessing DOS partitions

After the boot disk image has been created as described above, the DOS emulator can be booted, but there is no access to the Linux file system or to a DOS partition on the hard disk.

To access the Linux file system from within the emulator, use either the driver `emufs.sys` or the program `lredir`. The driver `emufs.sys` can map any directory of the Linux file system onto the next free drive letter. The redirector program `lredir` can replace already assigned drive letters. For example, `lredir` can be invoked as follows :

```
c:\>lredir c: \linux\fs/
```

The first parameter, in our example `c:`, specifies the drive letter to be replaced. The second parameter identifies the source, in our example the root directory of the Linux file system. `\linux\fs` represents the Linux file system and `/` specifies the path within the file system. The file `lredir.readme`, in the same directory as `lredir` program itself in the DOS emulator distribution, contains a detailed description of all the program's options.

There are many other ways to boot from a disk image, but they generally have negative consequences regarding subsequent access to floppy disk drives or other drive letters. We offer a brief overview of these approaches.

Assume that the hard disk contains a DOS partition that was mounted under Linux as directory `/dosd`. There are several ways to access this partition under the DOS emulator:

- Boot from a diskette and invoke the driver `emufs.sys` in the file `config.sys` or the program `lredir` in the file `autoexec.bat`. This makes the directory `/dosd` under Linux the next alphabetical drive in the DOS emulator. In this case this would be `C:`. Since it is rather troublesome to always insert a diskette to boot the emulator, this method is used seldom, and usually only during installation.
- Boot from a hard disk image and use `emufs.sys` or `lredir` as above to access drive `D:`. The hard disk image file uses `C:` in this case.
- Boot from a hard disk image and use `lredir` in the file `autoexec.bat` to replace the drive letter `C:`. This makes the DOS partition accessible as `C:` and the floppy disk drives remain untouched. This provides the same environment as booting directly from DOS.

The problem with this method is that the “hard disk” from which the file `autoexec.bat` is read is replaced by another while the file is still being read and executed. Thus this file needs to be identical in the hard disk image and on the real hard disk partition.

- Activate a disk image file as virtual drive `A:` and use the file `config.sys` or `autoexec.bat` as described above. Here the advantage is that you do not need a diskette to start the DOS emulator. The drawback is that the disk drive can no longer be used as drive `A:` because this letter is already occupied by the disk image.
- Use the special option `bootdisk` (see the start of this section) in the configuration file of the DOS emulator in order to boot from a disk image and switch this image off at the end of the file `autoexec.bat`. This is the most elegant method. Booting produces a normal environment without image files, yet a disk image can be used for booting.

Caution! It is theoretically possible to access an DOS partition on a hard disk both directly via the DOS emulator and indirectly via the Linux file system, assuming that the DOS file system is mounted. To prevent data loss, completely avoid this dual access. To access an DOS partition from both Linux and the emulator, the emulator should access the DOS partition not directly, but via the device driver `emufs.sys` or `lredir` and the Linux file system.

Configuration

The following example shows a complete configuration file for the DOS emulator in which the option `bootdisk` is used as described above. The example is divided into logical sections. Lines beginning with `#` are comments. The file is intended only as an example configuration, since it does not embrace all possible options.

```
# Example of a configuration file for the DOS emulator

# No debug messages
debug -a

# DOSemu Version message on startup
dosbanner on

# Enable access to video board ports
allowvideoportaccess on

# Keyboard and timer interrupts
keybint on
timint on

# Mouse options :
# "microsoft"(default), "mousesystems3," "mousesystems5,"
# "mmseries" or "logitech."
serial {mouse microsoft device /dev/cua0 }

#serial {mouse device /dev/cua0 }
#serial {mouse device /dev/cua1 }

# IPX in DOS-emulator (under development)
ipxsupport off

#video {vga console }
#video {vga console graphics chipset et4000 memsize 1024 }
#video {vga console graphics chipset trident memsize 1024 }
#video {vga console graphics chipset diamond }

# No special video supported, but graphic mode is ok.
video {vga console graphics }

# Direct access to keyboard
RawKeyboard

mathco off          # on or off
cpu 80386
boota               # booting from A:
xms 1024            # XMS size in K, or off
ems 1024            # EMS size in K, or off
```

```
# Here certain I/O ports can be allocated to the DOS-emulator
# ports {0x388 0x389 }
# ports {0x2f8 0x2f9 0x2fa 0x2fb 0x2fc 0x2fd 0x2fe }
# ports {0x21e 0x22e 0x23e 0x24e 0x25e 0x26e 0x27e 0x28e 0x29e
}

# Access to speaker
speaker native          # native, off or emulated

# Direct access to hard disks or hard disk images is set.
# In our example we use lredirect to access the partition
# /dev/hda1 --- no direct access

#disk {image "/usr/lib/dosemu/hdimage" }
#disk {partition "/dev/hda1" 1 readonly }
#disk {wholedisk "/dev/hda" }

# The file bdisk is used for booting. It is created from a
boot
# diskette with the following command:
# dd if=/dev/fd0 of=bdisk bs=16k
# At the end of the file autoexec.bat for this boot file,
# the file is disabled with the command bootoff and so access
# to drive A is restored.
bootdisk {heads 2 sectors 18 tracks 80 threeinch file bdisk }

# Direct access to floppy disk drives
floppy {device /dev/fd0 threeinch }
floppy {device /dev/fd1 fiveinch }

# This example uses no lasting disk image.
#floppy {heads 2 sectors 18 tracks 80 threeinch file bdisk }

#Printer access is not activated in this example.
#printer {options "%s" command "lpr" timeout 20 }
#printer {options '-p %s' command "lpr" timeout 10 }    # pr
format it
#printer {file "lpt3" }
```

The following debug settings are of interest only to developers and freaks. Normal users should set all debug options off. Debug settings

```
debug {config off disk off warning off hardware off
      port off read off general off IPC off
      video off write off xms off ems off
      serial off keyb off dpms off
      printer off mouse off
}
```

The dosemu startup message and the timer interrupt can be toggled on and off: startup message and timer

```
dosbanner on
#
# timint is necessary for many programs.
timint on
```

The keyboard also needs to be defined.

keyboard


```
# Possible values for keyboard layout:
#
#      finnish          us          dvorak          sf
#      finnish-latin1  uk          sg          sf-latin1
#      gr              dk          sg-latin1      es
#      gr-latin1       dk-latin1   fr          es-latin1
#      be              no          fr-latin1
portuguese
#
keyboard {layout gr-latin1 keybint on rawkeyboard on }
WOLLT IHR HIER EINE US-TASTATUR?? #
# After how many keyboard polls is the CPU released ?
# (0 turns this mechanismus completely off.)
HogThreshold 0
```

serial interfaces
and mouse

The settings for serial interface s, a serial mouse and a modem are optional. For a normal configuration, where the emulator runs under X11 and modem programs run directly under Linux, you can ignore these settings. To use the mouse under X11, you must not specify the mouse in the DOS emulation configuration file.

```
# activate one of the following settings as applicable:
#serial {com 1 device /dev/modem }
#serial {com 2 device /dev/modem }
#serial {com 3 device /dev/modem }
#serial {com 4 device /dev/modem }
#serial {com 3 base 0x03E8 irq 5 device /dev/cua2 }
#
# serial mouse:
#serial {mouse com 1 device /dev/mouse }
#serial {mouse com 2 device /dev/mouse }
#
# Type of mouse (do not use to run under X11!)
#mouse {microsoft }
#mouse {logitech }
#mouse {mmseries }
#mouse {mouseman }
#mouse {hitachi }
#mouse {mousesystems }
#mouse {busmouse }
#mouse {ps2 device /dev/mouse internaldriver }
```

New versions of the DOS emulator can also be operated dosemu on xterm remotely. Output goes to a normal xterm or color-xterm. However, the type of terminal and character set need to be defined.

```
***** TERMINALS *****
#
# IBM character set
#terminal {charset ibm color on method fast }
#
# color xterms or rxvt's without IBM font, 8 colors
#terminal {charset latin color xterm method fast }
#
# color xterms or rxvt's with IBM font, 8 colors
#terminal {charset ibm color xterm method fast }
#
```

```
# other terminals (not xterms or vt100)
#terminal {charset latin  color on  method ncurses }
#
# or further options ...
#terminal {charset latin  updatefreq 2  updatelines 25  color
on
#                method fast  corner on }
```

If the DOS emulator is to run under X11, then screen update frequency can be adapted. Furthermore, the X11 options such as the name of the window and the icon can be defined.

```
# possible keywords:
#  "updatefreq" value                (default 8)
#    Update frequency of screen output.
#    Smaller value = more frequent refreshes.
#
#  "updatelines" Wert                (default 25)
#    How many lines are updated
#
#  "display" string                  (default '')
#    Display X server
#
#  "title" string                    (default '')
#    Title of window
#
#  "icon_name" string                (default '')
#    Name of icon

X {updatefreq 8 updatelines 25 title "DOS in a BOX" icon_name
"xdos" }
```

To run the DOS emulator on a console, you need to specify the video adapter. This setting determines whether graphic programs that directly access the video adapter will function. Incorrect settings here can result in a black screen and also the screen remaining black after ending the emulator or switching virtual consoles. The only remedial action then is restarting the computer. To do this, blindly switch to another virtual console and shut down the computer with `shutdown -r now` (see section 7.3).

video adapter

reboot

```
***** video adapter *****
# This sets the video adapter. This setting is critical.
# Careless settings can disable the system. ...
#
# Under X11 (with dos -x or xdos) these settings are
irrelevant.
#
allowvideoportaccess on
#
# Standard VGA cards should function with this setting:
video {vga  console  graphics memsize 1024}
#
#video {vga  console  graphics  vbios_seg 0xe000 }
```

```
#video {vga console graphics chipset trident memsize 1024 }
#video {vga console graphics chipset diamond }
#video {vga console graphics chipset et4000 memsize 1024 }
#video {vga console graphics chipset s3 memsize 1024 }
```

Additional settings inform DOS which processor and coprocessor are installed. Further, the drive letter of the boot drive should be set to either A or C; our example employs `BootA`.

```
# Math coprocessor (on / off)
mathco on
# CPU type (286 / 386 / 486)
cpu 80386
# Boot drive (BootA or BootC)
bootA
```

Memory management within the emulator can be specified in relative detail. XMS, EMS and DPMS, as well RAM for adapters can be defined by size or switched off.

```
# XMS size in KB or off
xms 1024
# EMS size in KB or off
ems 1024
# For EMS, frame address can also be specified:
# ems {1024 ems_frame 0xe000 }
# ems {ems_size 2048 emsframe 0xd000 }
#
# Where should RAM be merged?
# (Make changes here only if you know what you are doing!)
# hardware_ram {0xc8000 range 0xcc000 0xcffff }
# DPMS size in KB or off
# DPMS might not be completely developed ...
dpms off
```

I/O ports and interrupts

Managing interrupts and I/O ports is a critical matter. The options are intended primarily for developers and freaks who are familiar with the internal workings of DOS. Ordinary users should make no changes here.

```
sillyint off
#sillyint {15 }
#sillyint {use_sigio 15 }
#sillyint {10 use_sigio range 3 5 }
# ports {0x388 0x389 } # for SimEarth
```

Access to PC speakers can be either direct or converted to beeps. speakers

```
# Access to PC speakers. Possible options are:
# native      direct access
# emulated    convert to beeps
# off
#
speaker native
```

We do not recommend direct access to hard disks and partitions. hard disk
However, if you have good reasons to do so, e.g., to access a partition compressed with a stacker, this access can be enabled here.

```
# hard-disk image
#disk {image "/var/lib/dosemu/hdimage" }
#
# direct access to first partition of hda with option
# read-only for write protection
#disk {partition "/dev/hda1" 1 readonly }
#
# direct access to the entire first AT bus disk (hda)
#disk {wholedisk "/dev/hda" }
```

The following option specifies that booting will take place from a disk image. boot image

```
#
bootdisk {heads 2 sectors 18 tracks 80 threeinch
file /var/lib/dosemu/bdisk }
```

Floppy disk drives need to be defined to allow access. diskettes

```
# Type and device for floppy disk drives
floppy {device /dev/fd0 threeinch }
floppy {device /dev/fd1 fiveinch }
```

Access to the printer can be redirected from the DOS emulator printer
to a corresponding device or even to a command under Linux.

```
#printer {options "%s" command "lpr" timeout 20 }
#printer {options '-p %s' command "lpr" timeout 10 }
#printer {file "lpt3" }
```

autoexec.bat

The `autoexec.bat` file which is used in the boot disk image could take the following form:

```
@echo off
lredir c: linux\fs/dosc
PATH C:\BAT;C:\TOOLS;C:\DOS;C:\EMU;C:\WINDOWS\WIN31;

lh DosKey
prompt $p$g
c:
bootoff
```

After execution of the file `autoexec.bat`, the boot file can no longer be accessed because the command `bootoff` disables the disk image and restores access to the real disk drive. To make later changes in the boot configuration, enable the boot file again with the command `booton`.

bootoff
changing boot
configuration

Observe that the file `autoexec.bat` !blank lines contains no blank lines after `bootoff`, since DOS would otherwise try to continue after the `bootoff` command. Since the file `autoexec.bat` no longer exists then, an error message would result.

blank lines

Accessing Novell servers

The DOS emulator can also access Novell servers. Since this area is under continuous development, the procedure also changes rapidly. Principally you can either configure the emulator so that it already provides an IPX interface, or you can start an adapted version of the `pdipx` driver, contained in the DOS emulator package, from the DOS emulator. Both cases start exclusively a normal `netxshell`. This establishes the connection to the server and provides a new virtual (Novell) drive. Then it suffices to change to the Novell drive and to launch the server's `login` command.

IPX

Current information on this procedure should be included in the `readme` file of the respective DOS emulator package.

readme

Starting and ending

Start the DOS emulator with the command `dos` or `xdos`. Here `xdos` is only a reference to `dos` that launches the X11 version of the emulator. Starting `dos` with the option `-x` has the same effect. Other possible options generally serve to overwrite options in the configuration file

xdos

-x

of the DOS emulator. Display these options by invoking `dos -help`. Normally you will not need the other options.

To terminate the DOS emulator, invoke the program `exitemu`, press the key combination `<Ctrl-Alt-BildAb>`, or simply end the process of the DOS emulator from another virtual console with `kill`.

-help

exitemu

4.2 WINE

A development that promises to have a significant influence on the dissemination of UNIX on PCs both in the private and the commercial sectors is the Windows Application Binary Interface (WABI).

WABI

This interface was developed by Sun Microsystems and licensed by numerous UNIX producers, so that it will be available on all common platforms. This interface makes it possible to run Microsoft Windows programs directly under UNIX. The interface converts Windows API calls to corresponding X11 calls, which has the interesting side effect that such programs can also be used by other X-servers due to the network transparency of X11.

Sun

MS Windows

X11

In early June 1993 the mailing list of the Linux DOS emulator began to host discussions about the porting or new development of a WABI for Linux. Within days a separate mailing list was set up for this project and the concrete development began.

DOS emulator

First there were two relatively independent developments. One group developed a loader for MS Windows programs, while another group began development of an API for converting MS Windows system calls to X11. The two groups soon agreed on a common course, and the result was today's WINE.

API

WINE

Although the large software packages for Windows do not run yet, smaller applications can be launched. The newest version of WINE can be transferred per FTP from the directory `/pub/linux/ALPHA/Wine/development` of the server `tsx-11.mit.edu`.

FTP

4.3 iBCS2 emulator

Development has progressed farther on the iBCS2 emulator. This emulator enables starting a number of commercial applications, such

iBCS2

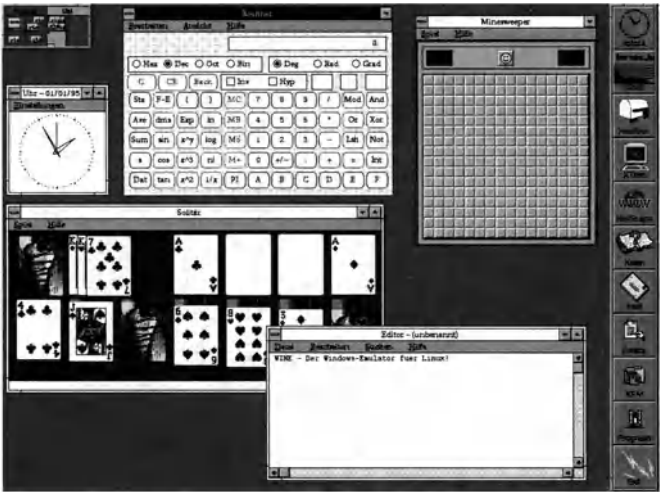


Figure 4.2. Solitaire under WINE

WordPerfect as WordPerfect, under Linux. The prerequisite is that the program must be available in one of the object formats employed by PC UNIX derivatives. UNIX System V Release 3 uses COFF; Release 4, ELF; SCO, COFF, ELF and SCO and interactive UNIX, a variant of COFF. Likewise Xenix V/386 and Wyse V/386 programs are in part executable under Linux.

SCO software packages are currently the most interesting because they are generally statically linked and thus require no additional shared libraries.

Installation

Since the iBCS2 emulator is available as a loadable module (see section 3.4), installation proves relatively simple. All you need to do is change to the unpacked directory with the source code and enter make. After compilation you have a new object file named iBCS, insmod which you load with insmod:

```
zeus:/root# insmod iBCS
```

It makes sense to enter the emulator in one of the startup scripts rc.local (rc.local) so that it is loaded on booting.

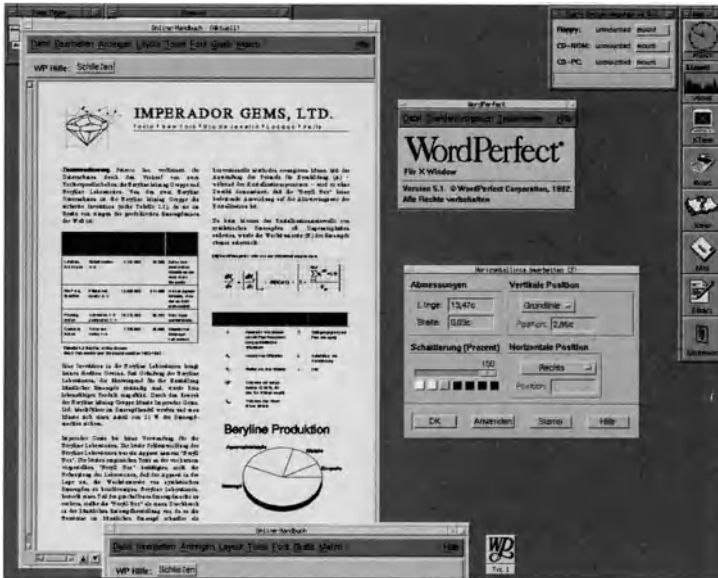


Figure 4.3. WordPerfect 5.1 under Linux

Complete installation requires you to create special device files. Observe the exact notation of the link `/dev/null` to `/dev/X0R` (X [zero] R).

```
zeus:/root# mknod /dev/socksys c 30 0 zeus:/root# ln -s
/dev/socksys /dev/nfsd
zeus:/root# ln -s /dev/null /dev/X0R
zeus:/root# mknod /dev/spx c 30 1
```

If you have a license for SCO shared libraries, copy these into a directory named `/shlib`. This permits execution of SCO programs that were not statically linked.

Applications

Thanks to the iBCS2 emulator, several interesting SCO applications can be used under Linux. The most prominent is certainly WordPerfect (version 5.1 or 6.0).

The Motif interface builder X-Designer has also been tested.

WordPerfect
X-Designer

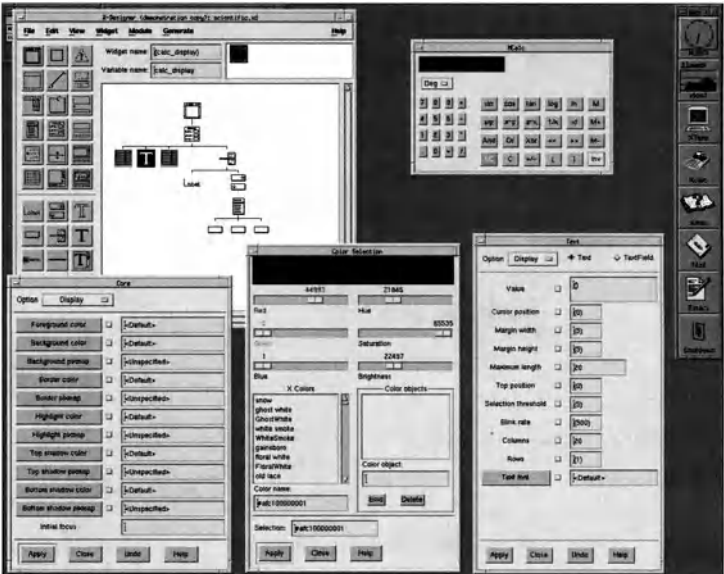


Figure 4.4. X-Designer

If you have the SCO shared libraries, then SCO Open Desktop 3 and various associated utilities run with no problem under Linux.

If you compare the performance of the applications under SCO and Linux, you will see that they run significantly faster under Linux. This is due to the more efficient implementation of the Linux kernel as well as the significantly faster X-server (XFree86).

4.4 HP48 emulator (X48)

Fans of the HP48 series will love the X48 emulator. It measures up to the original both optically and in functionality. For copyright reasons, operating the emulator additionally requires the contents of the original calculator's EPROMs. You can connect an external HP48 to the serial interface of the Linux computer. The main advantages of X48 over the original are significantly larger RAM and the possibility to cross-develop HP48 software. In addition to serious applications, this allows realization of computer games such as *Lemmings*.

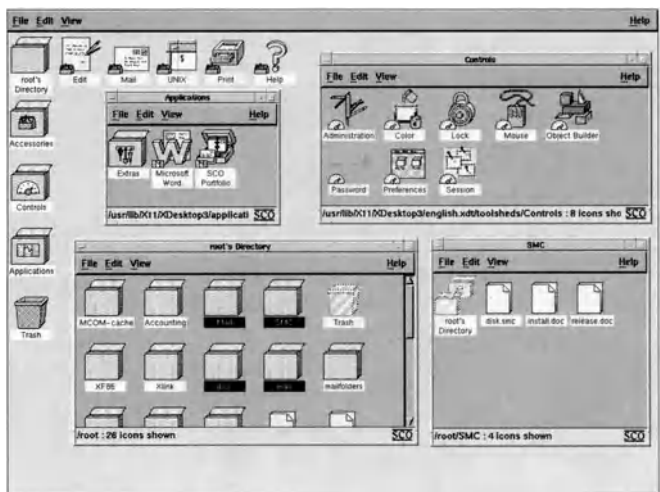


Figure 4.5. SCO's Open Desktop 3 under Linux



Figure 4.6. SCO ODT applications



Figure 4.7. HP48 emulator



Figure 4.8. IBM 3270 emulator under X11

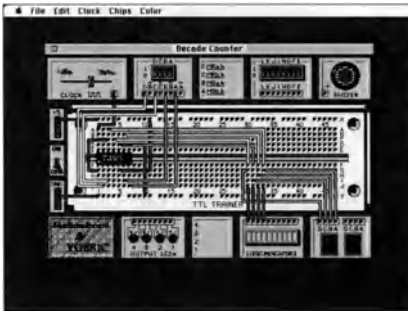


Figure 4.9. Apple Macintosh emulator

4.5 IBM 3270 emulator

The X3270 emulator allows Linux to connect to IBM mainframes. However, connection requires the availability of a TCP/IP stack on the respective mainframe. To attain perfect emulation of a 3270 terminal, you also need to install special character sets.

4.6 Macintosh emulator

An Apple Macintosh emulator has been presented by Abacus Research and Development. This is commercial software. Our impression from the demonstration version was quite positive.

Installation

This chapter imparts the most important sources for Linux and describes its installation. First we cover general features that apply to almost all Linux distributions. Then we explain the particular installation of the Slackware distribution.

5.1 Linux distributions

Since Linux, as free software, consists of many components that were and are being developed by different people all over the world, there is no *official* Linux installation package encompassing all the programs available for Linux.

official package

However, various groups and individuals offer their own packages consisting of the Linux kernel along with many utilities, applications and an installation program. Such a package is called a Linux distribution. Some producers of such distributions make it available on an FTP server package and also ship it on floppy disks or CD-ROMs for a fee. Linux distributions on FTP servers are normally spread over many subdirectories whose contents each fit nicely onto a floppy disk.

distribution

FTP server

Almost all such distributions include at least the kernel and all important utilities that you need for your initial installation. In addition, they usually include the GNU C compiler, the graphical user interface X11, and many other programs.

kernel

Linux installation is normally guided by a menu-driven installation program that permits selective installation of subsets of the distribution. Most installation programs also offer interactive configuration of the system.

installation program

In addition to installation from floppy disks, most distributions also support installation directly from CD, streamer tape or NFS from another computer. on your experience and the scope of the selected components, your installation time can range from one to several hours.

SLS

One of the first distributions to make a significant contribution to the propagation of Linux was SLS (Softlanding Systems) release by Peter MacDonald. For a long time it was the only distribution that enabled full installation with the C compiler and graphical user interface. With the appearance of other distributions such as Slackware, the SLS package declined in importance.

The current version used to reside in the directory `/pub/linux/packages/SLS` on the FTP server `tsx-11.mit.edu`. Meanwhile it has become hard to find.

MCC Interim

The University of Manchester assembled the MCC Interim release. Compared to other distributions, MCC is a lean package omitting less important programs such as X11 and TeX. The MCC distribution proves most suitable for users who want to install additional programs themselves and control exactly what is being deposited on their hard disks. However, we cannot recommend the package for novices.

Slackware

A newer distribution that enjoys relatively widespread propagation is the Slackware package. Originally based heavily on the SLS release, Slackware meanwhile uses its own installation routines that give a very mature impression. Slackware's color windows, dialog boxes and selection lists allow you at the beginning of the installation to specify the packages to be installed, the hard disk and partition to be used, and a country-specific keyboard table. The Slackware distribution permits installation from floppy disks, hard disk or CD or via NFS.

The scope of the Slackware distribution is quite extensive. In addition to the base system, X11, XView, applications, TCP/IP,

UUCP and network programs, Slackware offers a disk series with games and the newest GNU Emacs. Patrick Volkerding, manager of the Slackware package, assures that the package contains the current program versions, which certainly presents a challenge due to the rapid rate of development of the Linux system.

Patrick Volkerding

CD distributions

Many packages are now available on CD or as CD subscriptions. The subscriptions deliver a new CD several times a year with the newest version of the Linux kernel and several programs, or even a copy of a whole Linux directory tree from an FTP server. One well-known example is the CD from Yggdrasil Computing, which contains a bootable, installed Linux system as well as the source code of the MIT X11R6 system and many GNU utilities.

CD subscriptions

current programs

Yggdrasil

source codes

The CD distributions provide a simple and economical alternative for anyone who lacks an Internet connection. Some Linux CDs sell for under\$30, and a CD-ROM drive has become standard equipment on high-end PCs.

economical prices

The problem that such CD distributions pose, akin to diskette distributions sold by mail order or in software stores, is that Linux continues to evolve on a daily basis; in the weeks or months between the assembly of the distribution and production of the CD to the actual shipment, a new version with numerous improvements could already be available on the network.

production delay

Unifix

- manual The CD distribution from Unifix Software of Braunschweig, Germany, proves particularly interesting. In addition to the CD, the purchaser receives a small manual and a boot diskette that is used for the first system startup. After the installation of the 8 MB minimum configuration on an existing DOS partition or a new partition on the hard disk, the system is ready to use.
- direct from CD All application programs can be started directly from the CD. If the computer has enough memory (8-16 MB), then the most important data on the CD are retained in memory, which naturally has a very positive effect on the access time. Changing the CD suffices to make an update. The time-consuming installation of the complete Linux software becomes superfluous.
- software list The Unifix distribution also distinguishes itself from other distributions through its content. A software list, which can be read with a WWW client, contains a classification and overview of all available programs. The configuration of the included utilities and applications is mature and well thought-out.
- quality

5.2 Sources

- many sources With the escalating popularity of Linux recently, there are meanwhile many sources from which to obtain the package and additional programs for Linux. The most direct and quickest way to procure individual programs and kernel versions is certainly tapping into an FTP server on the Internet.
- FTP servers

FTP servers

- Finland The most important FTP server for Linux in Europe is `nic.funet.fi` in Finland, which was the original Linux server. The newest version of the kernel as well as any alpha releases reside here. MIT's FTP server `tsx-11.mit.edu` contains the most recent GNU C compiler and the Linux C libraries. A broad palette of Linux software can also be found at `sunsite.unc.edu` at the University of North Carolina. This server archives primarily software packages that have been ported to Linux.
- tsx-11
- C compiler
- sunsite
- programs

These servers are mirrored by several other FTP servers around the world. This means that the data on the FTP servers in Massachusetts and North Carolina are copied to the mirror servers at regular intervals. This is intended to reduce overall Internet traffic. (For European users, for example, a connection within Europe is usually faster and more reliable than direct access to the American servers.) The following are examples of such mirror servers around the world:

mirror servers

Textual name	Numeric address	Linux directory
tsx-11.mit.edu	18.172.1.2	/pub/linux
sunsite.unc.edu	152.2.22.81	/pub/Linux
ftp.funet.fi	128.214.248.6	/pub/OS/Linux
ftp.cdrom.com	192.153.46.2	/pub/linux
net.tamu.edu	128.194.177.1	/pub/linux
ftp.mcc.ac.uk	130.88.203.12	/pub/linux
src.doc.ic.ac.uk	146.169.2.1	/packages/linux
ftp.informatik.tu-muenchen.de	131.159.0.110	/pub/Linux
ftp.informatik.rwth-aachen.de	137.226.112.172	/pub/Linux
ftp.ibp.fr	132.227.60.2	/pub/linux
kirk.bond.edu.au	131.244.1.1	/pub/OS/Linux
ftp.uu.net	137.39.1.9	/systems/unix/linux
ftp.win.tue.nl	131.155.70.100	/pub/linux
ftp.stack.urc.tue.nl	131.155.2.71	/pub/linux
ftp.denet.dk	129.142.6.74	/pub/OS/Linux
nctucca.edu.tw	140.111.1.10	/Operating-Systems/Linux
nic.switch.ch	130.59.1.40	/mirror/linux
monu1.cc.monash.edu.au	130.194.1.101	/pub/linux
cnuce-arch.cnr.it	131.114.1.10	/pub/Linux
ftp.linux.org	198.182.196.129	/pub

The file `Linux FTP-Roadmap.table.z` roadmap provides an extensive overview of Linux FTP servers in Germany. This file resides, among other places, on the FTP server of the University of Erlangen (`FTP.uni-erlangen.de`) in directory `pub/Linux/LOCAL/Roadmaps`.

FTP roadmap

FTP mail servers

If you lack direct access to the Internet but can send and receive electronic mails, you have the option of reaching the Internet FTP servers by way of an FTP mail server. An FTP mail server can send e-mail you programs from FTP servers via e-mail. You simply send an e-mail with the corresponding command to the mail server, which then accesses the FTP server, splits the program into smaller parts, and sends these encoded with the UNIX command `uuencode` as an e-mail reply.

To obtain a list of commands that such an FTP mail server understands, the user sends a message with the command `help` in the first line to the FTP mail server. Examples of such FTP mail servers include `FTP-mailer@informatik.tu-muenchen.de` and `ftpmail@decwrl.dec.com`.

Please note, however, that such a server is not suitable for transferring a complete Linux distribution. Such servers are generally limited to transferring small amounts of data.

Commercial distributors

Other sources of Linux and Linux programs are the various commercial dealers who offer Linux diskettes and CDs in software stores or by mail order. Their addresses can be located in computer industry periodicals. Some of these dealers even offer hard disks and even complete PCs with a running Linux system already installed.

Mailboxes

Meanwhile in many larger cities mailboxes have sprung up that have specialized in Linux or at least offer some Linux programs. Matthias Gmelch has assembled a list that contains an overview of such Linux mailboxes around the world. This list can be found in the directory `/pub/linux/docs` on the FTP server `tsx-11.mit.edu` and its many mirror servers.

5.3 Hardware

One of the most frequently asked question regarding Linux is which hardware Linux supports and requires. The general prerequisite is a

PC-compatible computer with an 80386 or newer processor. Linux definitely does not run on old XTs or ATs with 80286 processors because Linux requires task-switching features that have only been available on PCs since the 80386.

386 PC or better

RAM

The minimal hardware configuration for Linux is an 80386 SX computer with 2 MB of RAM. The installation can become difficult because 2 MB of RAM is insufficient for installation programs. In this case a swap partition or a swap file must be created as soon as possible in order to allow launching programs and editors that are necessary for installation.

2 MB low end
swap partition

Normal work in text mode is possible starting at 4 MB of RAM. To be able to work with the X Window System in the normal version, 8 MB of RAM should be available. If the computer has 16 MB of RAM, this delivers a noticeable performance increase under X11.

8 MB RAM or more

Hard disk

By sacrificing the convenience of a graphical user interface in favor of a minimum hard disk installation, the storage requirements can be reduced to about 40 MB. However, for a full installation with X11, the C compiler, and all the tools and utilities, you need approximately 150 MB for Linux. There is hardly a ceiling on how much storage space a diligent network tapper can fill. With access to the Internet you will have no problem filling a 2 GB hard disk with free software. The supply of programming languages, utilities, libraries and application programs has reached enormous proportions.

150 MB
or more

For an economical initial installation, we recommend using a hard disk with an IDE or EIDE interface. These are available in many versions up to 1 GB and afford adequate performance in single-user mode. However, rather than a larger IDE or enhanced IDE hard disk and for professional use, we suggest going to a SCSI system.

IDE hard disk
single-user mode

The IDE interface was supported from the beginning by the Linux kernel and needs no additional driver. There is even a kernel patch that makes it possible to drive two IDE controllers in parallel as long as their I/O ports and interrupts are configurable.

two IDE controllers

For larger and faster hard disks, the Small Computer System Interface (SCSI) is usually used. It permits operating up to seven devices in parallel. These devices can be hard disk drives, streamer hard disk, streamer, CD-ROM tape drives, magneto-optical drives, CD-ROM drives, or scanners.

SCSI operation requires a special SCSI host adapter in the Linux kernel, but this is already contained in all current Linux distributions for the usual host adapters. Exotic host adapters should be avoided avoid exotics because drivers are seldom available. Drivers that are included with the device for DOS or other UNIX versions cannot be used with Linux.

Controllers by well-known companies such as Adaptec, NCR, Future Domain and Seagate pose no problems. In case of doubt, refer to the current list of supported hardware that is available on supported controllers HOWTO FTP servers under the title "Hardware HOWTO."

Video boards

An uncomplicated solution for a video board is to use a simple video board with an S3 chipset. The current X11 R6 package for Linux contains a special server optimized for this chipset. These boards are particularly fast in the 32-bit or 64-bit Local Bus or PCI variants of the S3 or Mach chipsets.

Many chipsets from other manufacturers are also supported. The current list of supported hardware can be found on FTP servers along with the other HOWTO s under the name "XFree86 HOWTO."

For an exotic video board or one equipped with little memory, the remaining option is the generic VGA server; however, it supports VGA mono only 16 colors and a resolution of 640 by 480 pixels, or the mono server, which also supports Hercules graphics.

Bus mice by all the familiar manufacturers, the usual serial mice mouse or a PS/2-compatible trackball can be used with Linux.

Bus system (ISA/EISA/PCI)

Linux supports mother boards with the old AT bus (ISA), with the more flexible and faster EISA bus, with Local Bus extensions, and with the newer PCI bus.

The PCI bus is processor-independent and significantly faster than the EISA system. It supports data transfer rates of up to 130 PCI bus

Mbytes/s in 32-bit mode. Several PCI mother boards have a very fast SCSI chip (NCR 53c810), for which the kernel contains a driver.

NCR SCSI chip

Although the specifications of the PCI standards were prepared with extreme care, some PCI components available today still deviate from it. This can cause compatibility problems between the mother board and peripheral boards. A dedicated PCI HOWTO provides more details on these difficulties as well as advice to potential purchasers of PCI hardware.

compatibility problems

Peripheral hardware

For data backup with streamers, Linux supports both SCSI devices and floppy streamers, of which we recommend the former.

streamers

The selection of drivers for network boards is quite rich. Besides the frequently-used boards from manufacturers like Novell, 3Com and SMC (who acquired the production of the old WD boards), several HP and DLink boards are supported. There are even drivers for Arcnet boards. Many generic boards are compatible to the above and can be used as well.

network boards

As is common in the UNIX tradition, Linux also allows the use of ASCII terminals that are connected to the serial port or to a special multiseriial adapter. Drivers are available for these boards as well.

multiseriial adapter

One feature that distinguishes Linux from many other systems is its direct kernel support of ISDN boards. The driver package `isdn4linux` enables using active ISDN boards as a network interface under Linux, without having to adapt the kernel. This software can be obtained via Internet from the following FTP server: `FTP.to.com:/pub/isdn4linux`.

ISDN

`isdn4linux`

Even multimedia applications are possible under Linux. Peripheral devices such as sound boards (Soundblaster, Soundblaster 16, Adlib, Gravis Ultra Sound or PAS 16) and CD-ROM drives can be operated with the corresponding drivers installed in the kernel. Here, too, it is important to check on the list of supported hardware to determine which CD-ROM drives currently have drivers. SCSI drives are particularly easy to connect.

sound boards
and CD-ROM

Even for exotic hardware such as transputer boards and frame grabber boards, drivers exist to enable operation under Linux.

transputers

recommended
hardware

Considering the continuing development on the hardware and software sectors, an 80486 computer with 33 MHz or higher and with 16 MB of RAM currently provides a reasonable and affordable configuration. The hard disk should have at least 200 MB of storage because a permanent shortage of storage precludes productive work. If you plan to use a CD-ROM drive or a streamer in addition to a hard disk, you should choose a SCSI configuration from the start.

14" monitor too small

A 14-inch monitor is really too small for a multitasking environment where multiple windows are usually open simultaneously. The Linux X server is able to provide a larger virtual console, and most devices can display 800 by 600 pixels reasonably. Thus for a start, a 17-inch monitor is not absolutely necessary.

5.4 Installation

Slackware

This section explains the Installation procedure in more detail using the Slackware distribution. This distribution seems to be the most widespread because it can be copied freely and is contained on many CD s from various vendors. With small deviations, this procedure applies to other distributions as well.

installation procedure

The basic steps of a Linux installation are:

1. Booting of a minimum Linux system from a boot (and possibly root) diskette
2. Creation of partitions for Linux
3. Creating file systems and swap regions
4. Copying the system to the hard disk
5. Configuration of the most important system files
6. Installation of a boot manager
7. Configuration of the graphical user interface
8. Creating the users

Boot diskette

booting from

The installation procedure begins with the booting of a minimum Linux system that contains only the kernel and only the most important utilities and the installation program. The Slackware

distribution contains two diskettes for this purpose, called the boot and the root diskette.

floppy disk
boot and root disk

Diskettes are provided in 3 $\frac{1}{4}$ -inch or 5 $\frac{1}{2}$ -inch versions. Both the boot diskette and the root diskette are available in various versions. The boot diskettes differ in the drivers that they contain in the kernel; the root diskette contains a minimal file system with the most important utilities and the installation script.

different versions
drivers

If the distribution was purchased as a diskette package, then the diskettes are preconfigured in a bootable version. Otherwise the installation diskettes contain image files. To create a bootable diskette or one with a Linux file system from such an image file, it must be transferred to the diskette with a special utility as described below. The images for boot and root diskettes are usually in the following directories of the Slackware distribution:

image files

- bootdsk.12
- bootdsk.144
- rootdsk.12
- rootdsk.144

The directory `bootdsk.144` contains the following files:

- README
- cdu535.gz
- old1118.gz
- WHICH.ONE
- loaded.gz
- sbpcd.gz
- alpha.gz
- mitsumi.gz
- scsi.gz
- bare.gz
- nec260.gz
- scsinet.gz
- cdu31a.gz
- net.gz
- xt.gz

For a 3 $\frac{1}{4}$ -inch floppy disk as DOS drive A is the boot drive, the directories `install/1.44meg/bootdsk` and `install/1.44meg/rootdsk` contain the correct data. The files for a 5 $\frac{1}{2}$ -inch floppy disk drive are found in the directory `install/1.2meg`.

3 $\frac{1}{4}$ or 5 $\frac{1}{2}$ inch

For remote installation of Linux via NFS, the kernel must contain a driver for the network board in the file for the boot diskette. If the computer has a network board for which the kernel on the boot

remote installation
drivers

diskette has no driver, you can create a custom boot diskette (see section 5.5).

SCSI host adapter The same applies to a system with a SCSI hard disk and an exotic SCSI host adapter for which none of the boot diskettes has the proper driver. In case of doubt, read the `README` file contained in the same directory as the disk image files, to determine which files are suitable for which hardware configurations.

root disk There are also different files for the root diskette, although here the distinction is only between a diskette with color support and a monochrome installation program. In most cases the file for the color installation program can be used: for a 3¼-inch drive, the file `rootdsks.144/color144.gz`. The other directories contain files for creating custom boot diskettes or additional tools and information.

creating floppy disks After you have selected the correct files for the boot and root diskettes, you need to transfer them onto formatted diskettes either under DOS with `rawrite.exe` or under UNIX with the `dd` command. First, decompress the source files with `gzip`. The following example demonstrates this procedure.

```
stef1:/tmp> gzip -d bare.gz
stef1:/tmp> dd if=bare of=/dev/fd0 bs=8k
180+0 records in
180+0 records out
stef1:/tmp>
```

booting After you have created the diskettes, insert the boot diskette in the boot drive and reboot the computer. The first message to be displayed comes from LILO, the Linux boot manager and loader, which is installed on the boot diskette. The following example shows the first message during booting.

```
Welcome to the Slackware Linux 2.1.0 bootkernel disk!

If you have any extra parameters to pass to the kernel, enter them at the
prompt below after one of the valid configuration names (ramdisk, mount,
drive2)

Here are some examples:

ramdisk hd=cyl,hds,secs      (Where "cyl," "hds," and "secs" are the number of
                               cylinders, sectors, and heads on the drive. Most
                               machines won't need this.)
```



```

In a pinch, you can boot your system with a command like:
mount root=/dev/hd1

On machines with low memory, you can use mount root=/dev/fd1 or
mount root=/dev/fd0 to install without a ramdisk. See LOWMEM.TXT for details.

If you would rather load the root/install disk from your second floppy drive:
drive2 (or even this: ramdisk root=/dev/fd1)

DON'T SWITCH ANY DISKS YET! This prompt is just for entering extra parameters.
If you don't need to enter any parameters, hit ENTER to continue.

boot:

```

The Linux Loader now waits for input. For a normal installation in which the boot drive is also used for the root diskette, it suffices to press **<Return>**.

In some cases, such as if a CD-ROM drive, a hard disk or a network board is not correctly recognized, special parameters have to be passed to the kernel. These parameters are listed after the specification of the boot option. An entry after the boot prompt has the following structure:

parameters

```

Boot choice Parameter=Value,Value,..
Parameter=Value,Value,...

```

An example was shown in the above Linux Loader message. The following is another example:

example network board

```
ramdisk ether=5,0x320
```

This tells the driver for the Ethernet board in the kernel that the board is using interrupt 5 and I/O address 0x320.

interrupt and IO address

If the choices are correct and any necessary options have been passed, then the Linux Loader loads the kernel. The kernel first decompresses itself and then initializes its individual components, displaying corresponding messages all the while. The messages of the device drivers indicate which devices were recognized and successfully initialized. The sequence of displays during the boot process could look like this:

kernel starts

device drivers

```

Loading ramdisk .....
Uncompressing linux...memory is tight...done.
Console: colour EGA+ 80x25, 1 virtual console (max 63)
bios32_init : BIOS32 Service Directory structure at 0x000fc300
bios32_init : BIOS32 Service Directory entry at 0xfc580
pcibios_init : PCI BIOS revision 2.00 entry at 0xfc5b0
Serial driver version 4.00 with no serial options enabled
tty00 at 0x03f8 (irq = 4) is a 16550A
tty01 at 0x02f8 (irq = 3) is a 16550A
lp_init: lp2 exists, using polling driver
Calibrating delay loop... ok - 36.08 BogoMips
scsi-ncr53c7,8xx : at PCI bus 0, device 4, function 0
scsi-ncr53c7,8xx : NCR53c810 at memory 0xfc800000, io 0xd000,
irq 11
scsi0 : using io mapped access
scsi0 : using initiator ID 7
scsi0 : using level active interrupts.
scsi0 ; burst length 8
scsi0 : using 40MHz SCSI clock
scsi0 : m_to_n = 0x90, n_to_m = 0xa0, n_to_n = 0xb0
scsi0 : NCR code relocated to 0x383b10
scsi0 : testing
scsi0 : test 1 started
scsi0 : tests complete.
scsi0 : NCR53c{7,8}xx (rel 3)
scsi : 1 hosts.
Vendor: TOSHIBA      Model: CD-ROM XM-3401TA  Rev: 2873
Type:   CD-ROM                      ANSI SCSI revision:
02
scsi : detected total.
Memory: 29780k/32768k available (868k kernel code, 384k
reserved, 1736k data)
This processor honours the WP bit even when in supervisor
mode. Good.
Floppy drive(s): fd0 is 1.44M
FDC 0 is a post-1991 82077
Swansea University Computer Society NET3.017
Swansea University Computer Society TCP/IP for NET3.017
IP Protocols: ICMP, UDP, TCP
eth0: SMC Ultra at 0x240, 00 00 C0 16 02 A0, IRQ 5 memory
0xe0000-0xe3fff.
smc-ultra.c:v1.10 9/23/94 Donald Becker
(becker@cesdis.gsfc.nasa.gov)
Checking 386/387 coupling... Ok, fpu using exception 16 error
reporting.
Checking 'hlt' instruction... Ok.
Linux version 1.1.59 (root@fuzzy) (gcc version 2.5.8) #6 Sat
Oct 29 1994
RAMDISK: 1474560 bytes, starting at 0x21b850

```

After the kernel has been successfully loaded and started, a switch floppy disk message appears on the screen that prompts the user to replace the boot diskette with a root diskette containing the basic file system.

```

Please remove the boot kernel disk from your floppy drive,
insert a
root/install disk (such as one of the Slackware color144,
colrlite,
tty144, or tty12 disks) or some other disk you wish to load
into a
ramdisk and boot, and then press ENTER to continue.

```

When the root diskette is ready, its contents are read into a RAM-disk, thus completely loading the root file system and the installation program into memory and freeing the floppy disk drive for other diskettes.

ramdisk

```
VFS: Disk change detected on device 2/28
RAMDISK: Minix filesystem found at block 0
RAMDISK: Loading 1440 blocks into RAM
disk.....
.....
done
```

On completion of booting, a welcome message appears on the screen that explains subsequent steps.

welcome

```
Welcome to the Slackware Linux installation disk, (v. 2.1.0)

##### IMPORTANT! READ THE INFORMATION BELOW CAREFULLY. #####
- You will need one or more partitions of type "Linux native" prepared.
It is
  also recommended that you create a swap partition (type "Linux swap")
prior
  to installation. Most users can use the Linux "fdisk" utility to
create and
  tag the types of all these partitions. OS/2 Boot Manager users,
however,
  should create their Linux partitions with OS/2 "fdisk," add the
bootable
  (root) partition to the Boot Manager menu, and then use the Linux
"fdisk" to
  tag the partitions as type "Linux native."
- If you have 4 megabytes or less of RAM, you MUST activate a swap
partition
  before running setup. After making the partition with fdisk, use:
  mkswap /dev/<partition> <number of blocks> ; swapon /dev/<partition>
- Once you have prepared the disk partitions for Linux, and activated a
swap
  partition if you need one, type "setup" to begin the installation
process.
- If you want the install program to use monochrome displays, type:
  TERM=vt100
  before you start "setup."

You may now login as "root."
slackware login:
```

To continue, enter the user name `root` at the login prompt. This brings another message:

log-in

```
Linux 1.1.59. (Posix).

If you're upgrading an existing Slackware system, you might
want to
remove old packages before you run 'setup' to install the new
ones. If
you don't, your system will still work but there might be some
old files
```

```
left laying around on your drive.

Just mount your Linux partitions under /mnt and type
'pkgtool'. If you
don't know how to mount your partitions, type 'pkgtool' and it
will tell
you how it's done.

To start the main installation, type 'setup'.
```

All subsequent steps take place under Linux.

Partitioning

multiple partitions
/home separate

Installing Linux requires at least one free partition. However, it makes sense to create three partitions: one for the root file system, one for the home file system, and one as a swap partition. Then a new installation of the system can leave the /home partition unscathed and ready for remounting after the installation. This also retains all user files.

fdisk

If the hard disk does not contain any partitions that can be used for Linux, then the partitions must be created first with the Linux command `fdisk`. The following example demonstrates the creation of partitions for Linux. Assume that one partition for DOS already exists on the hard disk.

Type of a partition

The partition type that is specified for a new partition is 83 (Linux native). The intended swap partition is set to type 82 (Linux swap).

DOS-fdisk

A partition's type is read by the installation program, which thus recognizes which partitions can be used for the installation. If partitions for Linux already exist, for example, which were created under DOS with `fdisk`, then the type of any partition to be used for a Linux file system should be set to 83 with the `fdisk` command under Linux.

fdisk under Linux

The following terminal capture reflects the execution of `fdisk` under Linux:

```
# fdisk
Using /dev/hda as default device!
Command (m for help): p
Disk /dev/hda: 14 heads, 35 sectors, 978 cylinders
Units = cylinders of 490 * 512 bytes

   Device Boot   Begin        Start    End  Blocks    Id  system
/dev/hda1             1            1    418   102392+    6   DOS
16-bit >=32M

Command (m for help): n
Command action
```

```

e   extended
p   primary partition (1-4)
p
Partition number (1-4): 2
First cylinder (419-978): 419
Last cylinder or +size or +sizeM or +sizeK (419-978): +100M

Command (m for help): p
Disk /dev/hda: 14 heads, 35 sectors, 978 cylinders
Units = cylinders of 490 * 512 bytes
   Device Boot   Begin        Start      End   Blocks   Id  system
/dev/hda1             1            1      418   102392+    6   DOS 16-bit
>=32M
/dev/hda2           419          419      836   102410    83   Linux
native

Command (m for help): n
Command action
  e   extended
  p   primary partition (1-4)
p
Partition number (1-4): 3
First cylinder (837-978): 837
Last cylinder or +size or +sizeM or +sizeK (837-978): +8M

Command (m for help): p
Disk /dev/hda: 14 heads, 35 sectors, 978 cylinders
Units = cylinders of 490 * 512 bytes
   Device Boot   Begin        Start      End   Blocks   Id  system
/dev/hda1             1            1      418   102392+    6   DOS
16-bit >=32M
/dev/hda2           419          419      836   102410    83   Linux
native
/dev/hda3           837          837      870    8330     83   Linux
native

Command (m for help): n
Command action
  e   extended
  p   primary partition (1-4)
p
Partition number (1-4): 4
First cylinder (871-978): 871
Last cylinder or +size or +sizeM or +sizeK (871-978): 978

Command (m for help): p
Disk /dev/hda: 14 heads, 35 sectors, 978 cylinders
Units = cylinders of 490 * 512 bytes
   Device Boot   Begin        Start      End   Blocks   Id  system
/dev/hda1             1            1      418   102392+    6   DOS
16-bit >=32M
/dev/hda2           419          419      836   102410    83   Linux
native
/dev/hda3           837          837      870    8330     83   Linux
native
/dev/hda4           871          871      978   26460     83   Linux
native

Command (m for help): t
Partition number (1-4): 3
Hex code (type L to list codes): L
 0 Empty                8 AIX                75 PC/IX                b8
BSDI swap
 1 DOS 12-bit FAT       9 AIX bootable      80 Old MINIX            c7
Syrinx
 2 XENIX root           a OPUS              81 Linux/MINIX          db
CP/M
 3 XENIX usr           40 Venix 80286       82 Linux swap           e1 DOS
access
 4 DOS 16-bit <32M     51 Novell?           83 Linux native         e3 DOS
R/O
 5 Extended            52 Microport        93 Amoeba               f2 DOS
secondary

```

```
6  DOS 16-bit >=32 63  GNU HURD      94  Amoeba BBT    ff  BBT
7  OS/2 HPFS      64  Novell       b7  BSDI fs
Hex code (type L to list codes): 82
Changed system type of partition 3 to 82 (Linux swap)

Command (m for help): p
Disk /dev/hda: 14 heads, 35 sectors, 978 cylinders
Units = cylinders of 490 * 512 bytes

   Device Boot   Begin    Start    End  Blocks   Id  system
/dev/hda1             1         1    418   102392+   6  DOS 16-bit
>=32M
/dev/hda2             419        419    836   102410   83  Linux
native
/dev/hda3             837        837    870    8330   82  Linux swap
/dev/hda4             871        871    978   26460   83  Linux
native

Command (m for help): w
The partition table has been altered!
Calling BLKRRPART ioctl() to re-read partition table
Syncing disks
Reboot your system to ensure partition table is updated
#
```

fdisk under Linux

All partitions, including the logical drives of an extended partition, are simply numbered sequentially under Linux and can be used like a primary partition. All partitions are mapped onto files in the directory `/dev`, their names beginning with `hd` for normal hard disks or `sd` for SCSI hard disks. More detailed information can be found in Section 2.6.static – use dynamic

Creating file systems

Before a partition is capable of storing files, a file system must be created on it. In general this is done by selecting the respective menu item in the installation script.

Alternatively, a file system can be created manually with the command `mkfs`. This command is only a front end that invokes the respective program to create the selected file system according to the file system type parameter (option `-t`) that is passed to it. For an Extended-2 file system, for example, this is the command `mke2fs`.

```
# mkfs -t ext2 -c /dev/hda4
```

The parameter `-c` activates the verification of the blocks on the hard disk that are used for the file system.

Creating a swap partition

Similarly, a swap partition is created with the command `mkswap`. In addition to the device, the number of blocks in the partition must be specified. This can be obtained easily by invoking the program `fdisk` with the option `-s`, which specifies the size of a partition in blocks.

mkswap

```
# fdisk -s /dev/hda3 8330
# mkswap /dev/hda3 8330
Setting up swappspace, size = 8523776 bytes
#
```

To activate the new swap partition, enter the command `swapon` followed by a partition parameter. Display the current state of main memory with the command `free`.

swapon

```
# swapon /dev/hda3
# free
      total      used      free      shared    buffers Mem:
Swap:   8324         0      8324         888      2300
```

For a machine with 8MB of RAM or more, the swap partition, like the file system, could be created and activated in the installation program. If you select the menu item `ADDSWAP`, the hard disk is searched for a partition of type `Linux swap`. If such a partition is found, the user can register it as the swap partition. Then this partition can be initialized with `mkswap` and activated with `swapon`.

RAM

type Linux swap

If the machine has 4 MB of less RAM, then a swap partition has to be created and activated before the invocation of the installation program. The available memory will not suffice otherwise.

4 MB or less

Copying to the hard disk

Once the swap partition and the file system are set up, the Linux system can be copied onto the hard disk by invoking the installation program via the command `setup`.

setup

The most important items in the main menu are the selection of the source medium, the selection of the target partition, the choice of the components to be installed, and the start of the actual installation.

source medium

target partition

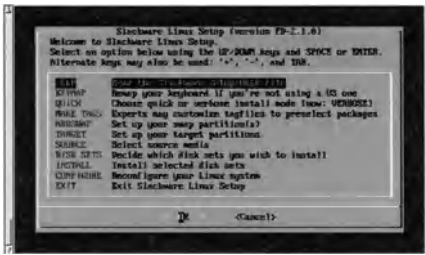


Figure 5.1. The main menu of the installation program

These menu items are automatically linked; that is, if the first point installation program is selected, the installation program automatically moves to the next item when the first item is complete. The choice of installation mode between quick mode and verbose mode should be left at verbose.

The source medium could be the hard disk if the files from the diskettes were copied there beforehand, the disk ettes themselves, hard disk, floppies, another computer on the network mounted with NFS, a CD-ROM or a streamer tape. Installation from a streamer tape is somewhat more tedious than the other possibilities and is explained in a separate README file .

NFS parameters An installation via NFS requires entering several network parameters such as the IP address of the target computer and that of the NFS server, the network mask, the broadcast and network address, and the path of the distribution on the NFS server. These terms are explained in Chapter 9. In case of doubt,

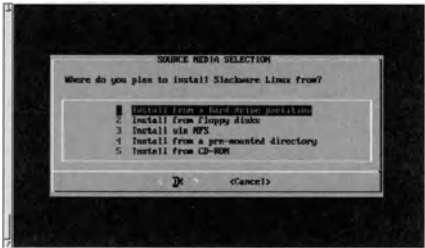


Figure 5.2. Selecting the source medium

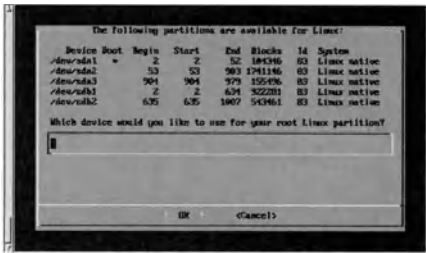


Figure 5.3. Selecting the target partition

call on the network administrator who knows the local setup for network administrator advice.

The selection of the target partition is also menu-driven. The installation program independently searches the system for all target partition partitions of type `Linux native` and displays these for selection:

After the root partition has been created, the installation program asks whether additional partitions are to be mounted. If you want to additional partitions install `/home` and possibly `/usr` on separate file systems, you need `/home` to specify these here.

Likewise you select the packages you want to install from a list of all possible packages of the distribution, accompanied by short packages descriptions.

Next you invoke the menu item `INSTALL` mode. You will be presented with a selection of modes in which the copying procedure can be carried out. For an initial installation we recommend the `NORMAL` mode mode as the best choice. This mode installs the descriptions

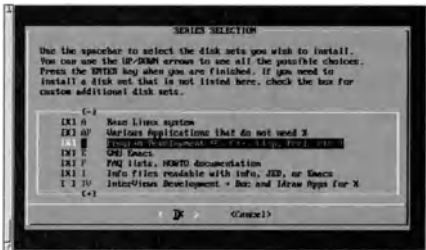


Figure 5.4. Selecting packages

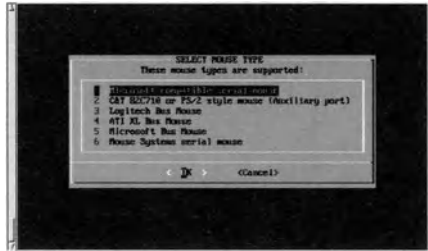


Figure 5.5. Mouse configuration

configurations fundamental system components automatically and asks the user about optional elements. Before each copying step the installation program displays information about the subpackage being installed or the option for selection, which eases deciding about optional components.

Once the selected packages are installed, the configuration of the system begins. Here various parameters and links are set, such as for the connection of a modem or a mouse.

LILO You also have the option of installing the Linux Loader in menu-driven mode.

On completion of the installation, the program automatically returns to the main menu, where you can quit. This ends the installation of the Slackware distribution and the computer can be rebooted.

5.5 Creating a boot diskette

SCSI If installing the Linux system requires a special driver in the boot kernel, for example, to access a SCSI hard disk drive with a rare host adapter or an exotic network board, then you can create a custom boot diskette. The prerequisite is naturally that the driver exists at all. Furthermore, you need access to a running Linux system in order to compile an appropriate kernel. The next chapter explains the configuration and compilation of the kernel (see section 6.2), so we only offer a concise description here.

compiling a kernel First you need to construct a kernel that contains all the necessary drivers. Drivers for peripherals that are not present should not be

compiled into the kernel because they increase the risk of conflict between drivers. In the directory with the Linux source code, invoke the command `make config`.

conflicts
make config
make dep

Now the kernel can be recompiled. First the command `make dep` determines the dependencies of the individual source code files of the kernel. Then any old object files that are still present are deleted with `make clean`. Start the actual compilation of the kernel with `make zImage` in the main directory of the kernel source code, i.e., `/usr/src/linux`. The result is a kernel named `zImage` located in the directory `/usr/src/linux/arch/i386/boot`. The compilation process takes from fifteen minutes to an hour, depending on the hardware.

make clean
make

```
zeus:/root# cd /usr/src/linux
zeus:/usr/src/linux# make dep
...
zeus:/usr/src/linux# make clean
...
zeus:/usr/src/linux# make zImage
...
```

Now some kernel parameters need to be modified with the command `rdev`.

kernel parameters
`rdev`

```
zeus:/usr/src/linux# rdev zImage /dev/fd0
```

This specifies that the root file system is on the diskette.

root file system

```
zeus:/usr/src/linux# rdev -R zImage 0
```

This declares that the root file system is to be mounted as read/write (writable).

```
zeus:/usr/src/linux# rdev -r zImage 1440
```

This sets the size of the ramdisk at 1.44 MB (also see section 6.2), which corresponds to a 3¼-inch, high-density diskette. To use a 5½-inch drive, substitute the number 1200 for 1440 above.

size of ramdisk

```
zeus:/usr/src/linux# rdev -v zImage -1
```

video mode

This assures that the kernel initializes the video board in normal mode (80x25).

boot disk
copy of another

Now you need to make a new boot diskette with this kernel image. The easiest way is to make a copy of another Slackware boot diskette (in our example we create a new boot diskette), mount it and copy the new kernel, named `vmlinuz`, onto it. Afterwards the diskette is unmounted from the file system with the command `umount`.

```
stef1:/usr/src/linux# zcat net.gz |dd bs=8192 of=/dev/fd0
0+360 records in
0+360 records out
stef1:/usr/src/linux# mount /dev/fd0 /mnt
stef1:/usr/src/linux# cat arch/i386/boot/zImage > /mnt/vmlinuz
stef1:/usr/src/linux# umount /dev/fd0
```

additional floppy disk

Now we need another diskette to which we will directly write a new kernel where the ramdisk is deactivated.

```
stef1:/usr/src/linux# rdev -r zImage 0
stef1:/usr/src/linux# cat zImage > /dev/fd0
```

change boot disk

We can now boot from this diskette. As soon as the following message appears, the boot diskette (the first) can be re-inserted:

```
VFS: Insert root floppy and press ENTER
```

change disks

After you press the **<Return>** key, the boot procedure continues. Several error messages might appear on the screen, but we will ignore them at this point.

After a short time the login prompt appears. Here you must log in as `root`.

```
(none) login: root
```

Next enter the commands `lilo` and `sync`.

`lilo`, `sync`

```
# lilo
Added ramdisk
Added drive2
Added mount
# sync
```

As soon as the `sync` command has executed and the prompt appears, you can remove the diskette from the drive. It is now a new boot diskette for the Slackware distribution.

5.6 Boot manager

The Linux Loader (LILO) permits loading Linux immediately on booting. If there are multiple operating systems on the hard disk, then LILO can manage the selection of the system to be started. Besides Linux, LILO can load DOS, OS/2 or a different PC-UNIX variant, and can even boot these from a second hard disk.

Linux Loader

selecting the OS

The operating principle resembles that of the OS/2 boot manager. Instead of loading an operating system immediately, LILO starts first and lets you choose from among all registered operating systems and configurations.

boot manager

Operation

The loader first presents itself with the word “LILO ” across the screen. Then you have a configurable amount of time to press one of the keys `<Alt>`, `<Ctrl>` or `<AltGr>` to not boot the standard configuration, but to select another partition or configuration. Then the loader prompts you with the configurable `boot :` message to enter a boot variant.

`<Alt>`, `<Ctrl>` or `<AltGr>`

As soon as the boot prompt appears, you can press the tabulator key to display a list of available alternatives.

`<Tab>`

```
LILO boot:
linux      linux-old      dos
boot: linux
Loading linux
```

automatic selection

If one of the above keys is not pressed during the preset time, then the first selection is made automatically and the first defined operating system is loaded.

Configuration

/etc/lilo.conf

We describe a typical installation for a system with an (E)IDE hard disk drive with the Linux root file system installed on the second partition. First adapt the file /etc/lilo.conf:

```
boot = /dev/hda2
root = /dev/hda2
install = /boot/boot.b
message = /boot/message
map = /boot/map
delay = 100
compact
image = /vmlinuz
        label = linux
        read-only
image = /vmlinuz.old
        label = linux-old
other = /dev/hda1
        label = DOS
```

MBR or start
of a partition

The first line specifies where LILO will be installed. The choices are at the start of a hard disk drive and thus in the master boot record (MBR), or at the start of a partition. The securest solution is to install LILO at the start of the Linux root file system and to activate this partition (see below).

Installing LILO in the master boot record (MBR) means overwriting the original DOS MBR. However, this can be restored with `fdisk /mbr` under DOS (or inadvertently with some virus programs).

root file system

`root = /dev/hda2` specifies where the kernel should attempt to mount the root file system.

`compact` optimizes LILO to prevent reading each sector individually.

boot sector
message

`install = /boot/boot.b` is an optional setting that specifies the boot sector to be installed. The default setting is `/boot/boot.b`.
`message = /boot/message` is likewise optional and specifies the name of the text file that is displayed on booting before the prompt. This file normally contains notes on the possible boot options and image files.

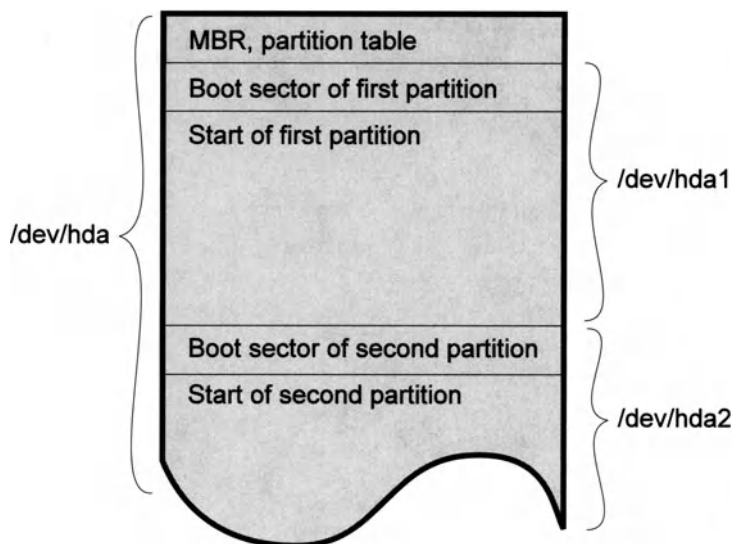


Figure 5.6. Logical structure of a hard disk

You can define `map = /boot/map` to specify another map file. Here during installation LILO stores information about the possible image files and options. The default value is `/boot/map`.

map file

`delay` specifies the time interval in tenths of seconds that LILO should wait for a key to be pressed before automatically selecting the first boot alternative.

delay

The entry `image = /vmlinuz` and the subsequent indented lines describe the first boot alternative. The kernel image file to be booted is `vmlinuz` and resides in the root directory. The designation that you need to enter to boot is `linux`. `read-only` overwrites the kernel settings that specify whether the root file system is mounted read/write or read-only.

kernel image files

read-only

The second boot alternative is the file `/vmlinuz.old`. This is an older kernel image file that has been saved for the event that the current kernel might cease to function.

other image files

The third boot alternative is the DOS partition on the first IDE hard disk. The tag `other` assures that LILO does not attempt to start Linux from this partition, but instead the loader of another operating system.

DOS

parameters for
kernel or init

A quite useful option in the LILO configuration file is `append`. After this option, as with the `boot` prompt of LILO, you can pass parameters for the kernel or the `init` process. This is often needed for sound boards or systems with two Ethernet adapters. The following example shows the file `/etc/lilo.conf` of a Linux computer that is used as a router and thus has two Ethernet boards.

```
boot    = /dev/sda1
root    = /dev/sda1
compact
delay   = 50
append  = "ether=10,0x280,eth0 ether=5,0x340,eth1"

image = /vmlinuz
        read-only
        label = linux

image = /vmlinuz.fw
        label = firewall
```

eth0 and eth1

The `append` line establishes the mapping of the Ethernet boards to the devices `eth0` and `eth1` as well as their interrupt and I/O port settings. See section 6.2 for an overview of possible kernel parameters.

Kernel Image Files

compiled kernel

A kernel image is a file that contains the actual kernel of the operating system along with an initialization program and loader. This file is created on compilation of the kernel whose source code resides in `/usr/src/linux` (see section 6.2).

LILO

To be able to boot from a kernel image file, you need to enter its name in the LILO configuration file and reinstall the loader.

old kernel image

After compiling a new kernel, you should also enter the old version of the kernel image in the configuration file. In case you made an error in the compilation or configuration of the kernel and the system does not boot with the new kernel, you can boot with the old kernel and correct the error.

make zliilo

This backup approach is already supported by the makefile of the kernel. On invocation of `make zliilo zliilo` in the kernel source code directory `/usr/src/linux`, after compilation the new kernel image file is copied to `/vmlinuz` and the old one is renamed to `/vmlinuz.old`.

Activating the loader

To activate the entries in the configuration files, LILO must be LILO reinstalled. Invoke the script `install` in directory `/etc/lilo`, which writes the actual Loader with its current configuration to a boot sector or the MBR.

```
dirkl:/root# lilo
Added linux
Added linux-old
Added DOS
dirkl:/root#
```

Removing LILO

If LILO was installed as described above in the partition of the Linux root file system, then reactivating the previously active partition activate old partition `fdisk`, e.g., to boot DOS directly, suffices to uninstall LILO.

If LILO was installed in the MBR and should now be removed for whatever reason, then the previous contents of the MBR must be restored. If DOS is available on another hard disk or partition, then the simplest solution is to invoke the DOS `fdisk` program with the fdisk /mbr option `/mbr`. This installs a new master boot record and overwrites LILO.

Alternative boot manager

To install a boot manager other than LILO, such as the OS/2 boot manager, then LILO must not be installed in the MBR; otherwise it OS/2 boot manager will definitely be started as the first boot alternative. .

The OS/2 version of `fdisk` should be used to set up the OS/2 fdisk partitions. Afterwards the OS/2 boot manager can be initialized and the boot partition can be activated. Then the Linux partition where the Linux Loader was installed should be registered as well. On booting the OS/2 boot manager is activated; from here the user can either select OS/2 or, via LILO, start the Linux system. Check the LILO User's Guide or the LILO FAQ for more information on the Linux Loader, or other FAQs and HOWTOs that are available on various FTP servers.

Configuration

Once the system files have been copied onto the hard disk and the Linux Loader has been installed, some of the configuration files need to be adapted. This fine-tunes the operating system to the available hardware.

6.1 General configuration

Most configuration files belong in the directory `/etc`. Many of these files are fixed in content and normally need not be changed. The appendix to this book contains a brief description of the files. This chapter discusses the configuration modifications that have to be carried out after the Linux Loader is installed on normal PC hardware.

File systems

When the system starts up, one of the `rc` scripts (see section 7.2) invokes the command `mount`, which mounts all file systems. The command is usually invoked with the option `-a`, which automatically mounts all file systems specified in the file `/etc/fstab` into the directory tree. This file should list all available file systems, including those that are not intended for automatic mounting. Files that are not to be automatically mounted are listed with the option `noauto`, which overrides automatic mounting with `mount -a`. This permits global specification of mount options for all file systems.

The following example (excerpt from the file `/etc/fstab`) shows these entries for a computer on which not only `/home` but also `/usr` and `/var` were installed on separate partitions:

/dev/sda1	/	ext2	defaults
/dev/sda2	/usr	ext2	defaults
/dev/sda3	/var	ext2	defaults
/dev/sda5	/home	ext2	defaults
/dev/sda6	none	swap	
none	/proc	proc	defaults
/dev/scd0	/cdrom	iso9660	defaults,user,noauto,ro
zeus:/usr/src	/usr/src	nfs	defaults

- device** The entries correspond largely to the parameters of the `mount` command. The first column identifies the device to be mounted. For an NFS mount, this is the host followed by a colon and the corresponding directory. The second column identifies the path of the *mountpoint*, the existing directory to which the new file system is to be mounted. The third column specifies the type of the file system.
- mountpoint**
- options** Most file systems permit the specification of additional options that are passed via the last column. If you do not want any particular parameters, you should enter `defaults` here to activate the standard options. Possible mount options are listed on page 390 in the Reference section.
- defaults**
- /proc** The entry that mounts the `/proc` file system is important because several commands, such as a variant of `ps`, are dependent on this directory. System information is represented here in the form of files and subdirectories.

Swap space

- RAM** If the target computer has 8 MB or less of RAM, it is necessary to create a swap partition or a swap file if this was not already done during installation. Otherwise the machine will not have enough memory to execute multiple programs simultaneously under the graphical user interface. Four MB of memory does not even suffice to recompile the kernel while running an editor at the same time. Even with 16 MB we recommend extending virtual memory with a swap partition.
- swapon** The command `swapon` activates the swap space (see section 5.4). Entering the swap partition in the file `/etc/fstab` like a file system affects its automatic activation on system startup. The preceding example contains such an entry.

In some cases it makes sense to use a swap file instead of a swap partition. Such a file must first be created in the desired size using the command `dd`. The following example demonstrates creating and activating a swap file:

swap file

```
hermes:/# free
      total      used      free      shared    buffers
Mem:    31380     29476     1904     17260     10092
Swap:      0         0         0
hermes:/# dd if=/dev/zero of=/swapfile bs=1k count=8192
8192+0 records in
8192+0 records out
hermes:/# mkswap /swapfile 8192
Setting up swapspace, size = 8384512 bytes
hermes:/# sync
hermes:/# swapon /swapfile
hermes:/# free
      total      used      free      shared    buffers
Mem:    31380     29484     1896     17260     10092
Swap:    8188         0     8188
hermes:/#
```

Swap files are also entered in the file `/etc/fstab` so that they can be automatically activated with the command `swapon -a`. Here the swap file is entered in place of the device:

/etc/fstab

```
/swapfile      none      swap
```

Login

If the shadow password package included in many distributions has been installed, the many options concerning user login are set in the file `login.defs` in the `/etc` directory. For example, this file specifies how long the login prompt is disabled (the delay) after the entry of an incorrect password and lists the devices from which the user `root` can log in.

shadow password

Corresponding notes can be found in the file `/etc/login.defs` itself or in the corresponding on-line Manual page. The following is a small excerpt from this file.

/etc/login.defs

```
# Delay in seconds for which the login prompt is disabled
# after entry of a wrong password.
FAIL_DELAY 2

# Devices from which a login as root is permissible
#CONSOLE    /etc/terminals
#CONSOLE    console:tty01:tty02:tty03:tty04
CONSOLE     tty1:tty2:tty3:tty4:tty5:tty6:tty8
```

```
# Files to be output after a login
MOTD_FILE /etc/motd
```

Excerpt from the file `/etc/login.defs`

other login packages

Alternative login packages that do not support shadow passwords normally do not include this file. They set the corresponding parameters at compilation.

Keyboard layout configuration

loadkeys

gr-latin1.map

As of Version 0.99.10 of the Linux kernel, keyboard settings are no longer bound to the compilation of the kernel; they can be changed at run time. The command `loadkeys` serves this purpose by loading a table containing a keyboard layout. On system startup, this command should be invoked from one of the `rc` files. For example, German-speaking users would use the file `gr-latin1.map` as their keyboard layout; it provides a normal German keyboard with umlauts as defined by ISO Latin-1. Similar keyboard maps exist for Spanish, French, and other languages.

`/usr/lib/kbd/keymaps`

To ensure comfortable operation under Linux, the appropriate country-specific keyboard map file needs to be chosen from the directory `/usr/lib/keymaps`, `/usr/lib/kbd/keymaps`, or `/etc/keymaps`, depending on the distribution. To load the correct keyboard layout on booting, invoke the command `loadkeys` in one of the `rc` files. The following excerpt from the file `/etc/rc.d/rc.local` demonstrates loading the German keyboard layout:

```
# loading country-specific keyboard layout for German
/usr/bin/loadkeys /usr/lib/keymaps/gr-latin1.map
```

6.2 Kernel

source code

Like the commands, utilities, and all other Linux programs, the kernel—the actual heart of Linux—is available as source code for free. The kernel was written primarily in C, with some small parts in Assembler. In addition to the scheduler, which manages the

switching between running processes, the kernel contains the drivers for peripheral devices and the routines for managing the file systems.

drivers

Configuration of the kernel

Another step after the base installation of the operating system is the configuration and compilation of the kernel. However, in many cases this can be omitted. In order to achieve optimal fine-tuning to the available hardware, compilation is absolutely recommended. This allows the administrator to dispense with drivers that are not needed, to add new drivers, and to modify settings of drivers. This customizing reduces the memory requirements of the kernel and accelerates booting.

fine-tuning

memory requirements

Normally the source code of the kernel is located in the directory `/usr/src/linux`. This directory also contains a configuration script that is invoked by the makefile and significantly simplifies the customizing of the kernel. This script is activated with `make config` and permits setting the following options:

`/usr/src/linux`

`make config`

- Supported file systems
- TCP/IP support and options
- Use of coprocessor emulator
- Optimization for 80486 processors
- SCSI support
- Drivers for SCSI and network boards
- Parameters for specialized interface boards
- Settings for sound boards

If the kernel fails to recognize special parameters such as the I/O address or interrupt number of a board and these need to be specified, it sometimes does not make sense to change these in the source code of the respective driver. Instead, they should be passed to the kernel on booting. Such parameters can be entered in LILO manually or as an `append` instruction in the file `/etc/lilo.conf` (see section 5.6). For example, a computer equipped with two Ethernet boards for use as a router or gateway must have the specifications of the boards entered, since the kernel normally expects only one network board.

I/O addresses
interrupts

`append`

Compilation

Once all necessary specifications have been made, the dependencies of the kernel among the individual parts of the source code have to be determined anew. This process is started with the invocation of `make dep`. This requires being in the main directory of the kernel source code (`/usr/src/linux`). Since old object files could still be present when the configuration is modified, these should first be removed with `make clean`.

At this point the actual compilation process can start. Here the `makefile` plays a central role, as during installation and compilation of most other programs: it contains the dependency information on the individual source code files and helps to coordinate various scripts for configuration and installation of the kernel.

Normally a compressed kernel is generated; then an integrated routine decompresses the kernel during booting. This feature of the Linux kernel makes it possible to put a minimal but complete Linux system that supports all hardware (network, streamer, SCSI devices, CD-ROM) onto a single boot disk. The time lost in decompression is insignificant.

The following lists the most important variants in the invocation of the `make` command:

- | | |
|-----------------------|---|
| dependencies | • make dep —Determines anew the dependencies among source code files, which should be done after any modification in the configuration of the kernel. |
| deleting | • make clean —Deletes all object files. On the next compilation, all source code files are compiled from scratch. |
| compilation | • make —Compiles kernel only; no kernel image is created. |
| kernel image | • make zImage —Compiles the kernel and creates a compressed, bootable kernel image in the directory <code>/usr/src/linux/arch/i386/boot</code> . |
| old image
/vmlinuz | • make zli10 —Creates a new kernel image as above and copies it to <code>/vmlinuz</code> . If an old file of the same name is present, it is stored to <code>/vmlinuz.old</code> . Then the installation script of the Linux Loader is invoked, so that with the next system startup the new kernel can be booted. |

- **make disk** — Writes the image file directly to the disk currently in drive A, on completion of compilation. Before this command is invoked, an empty, formatted diskette should be placed in drive A. boot disk

Configuration with rdev

Various parameters of the kernel can be set after compilation in the image file. This includes the device of the root partition and the size of the ramdisk. The program for making such settings is `rdev`. On invocation with the option `-help`, the program displays a list of all options. The most important invocations of the program are the following:

- **rdev -help** This command displays an overview of all possible options. help
- **rdev <image file> <Root-device>** e.g., `rdev /vmlinuz /dev/hda1` This invocation specifies the device from which the root file system is to be mounted on booting of the specified kernel image file. If the specified image file is loaded by LILO, then this configuration is usually overwritten by the LILO configuration. An entry of the form `root=device` in the file `/etc/lilo.conf` thus overrides a direct entry in the kernel. However, this setting in the kernel is important if the image file is written directly onto a disk with the command `dd`. root file system
- **rdev <image file>** e.g., `rdev /dev/fd0` This displays the device that is set in the image file as the device of the root file system. display
- **rdev -R <image file> <Flag>** e.g., `rdev -R /vmlinuz 1` This option determines whether the kernel mounts the root partition as read-only or as read/write. Many Linux distributions assume that the kernel first mounts the root partition as read-only, so that the file systems can be checked with `fsck` on booting. After the check, the file systems are remounted as read/write. If the kernel does not mount the root partition as read-only on booting, then no file system check is possible. read-only or read/write

Value 1 indicates read-only mode; value 0 is read/write.

- **ramdisk** `rdev -r <image file> <ramdiskSize>` e.g., `rdev -r /vmlinuz 1440` Here the size of the ramdisk can be set in the kernel. This option is used primarily for boot disks, where the device of the root partition is set to `/dev/fd0` and the size of the ramdisk to 1440, corresponding to a 3 $\frac{1}{4}$ " diskette. On booting, the kernel then loads the contents of the boot disk into the ramdisk and mounts this as the root file system. This frees the disk drive for other disks.
- **video mode** `rdev -v <image file> <VideoMode>` e.g., `rdev -v /vmlinuz -1` This sets the video mode with which the kernel initializes the video board. `-1` represents normal resolution; with `-3` the kernel prompts the user for the resolution on startup.

Passing parameters to the kernel

append The options of the various drivers can be passed to the kernel on booting. These parameters must be either entered manually after the boot prompt or specified in the configuration file of the Linux Loader in an append line (see section 5.6). The most important parameters are listed in the following overview:

- **root=device** specifies from which device the kernel mounts the root file system.
- **ro** defines that the root file system is to be mounted read-only.
- **rw** defines that the root file system is to be mounted read/write.
- **debug** sets the debug level within the kernel to 10.
- **no-hlt** switches off the invocation of the `hlt` statement in the kernel's idle loop.
- **no387** specifies that the kernel should not use the arithmetic coprocessor but its own emulation, which must be compiled in the kernel.
- **reserve=IO-addr, length, IO-addr, length...** prevents device drivers from accessing the specified range of I/O ports to independently find supported adapters on booting.
- **ramdisk=kilobytes** sets the size of the ramdisk to the specified value in kilobytes.
- **ether=IRQ, IO-addr, P1, P2, device** passes parameters of the installed Ethernet boards to the corresponding drivers in

the kernel. The meaning of **P1** and **P2** depends on the respective driver. Usually 0 is specified here. **device** contains the name of the Ethernet device to which the parameters apply. The default is `eth0`.

- **hd=cylinders, heads, sectors** defines the geometry of the hard disk. This option is only required if a hard disk is not recognized.
- **st=bufferSize, WT, MaxTapeBuffer** sets parameters for a SCSI tape drive. The exact meaning of the parameters can be found in the `README.st` file in directory `drivers/scsi` of the kernel source code.
- **bmouse=IRQ** sets the IRQ for the bus mouse.
- **max_scsi_luns=No** sets the highest logical unit number for SCSI devices.
- **st0x=ROM-Adresse, IRQ** sets parameters for Seagate st01 and st02 host adapters.
- **tmc8xx=ROM-addr, IRQ** sets parameters for Future Domain TMC8xx host adapter.
- **t128=ROM-addr, IRQ** sets parameters for T128 host adapter.
- **pas16=IO-addr, IRQ** sets parameters for the PAS 16 host adapter.
- **ncr5380=IO-addr, IRQ, DMA** defines parameters for a SCSI-host adapter with NCR 5380 chip.
- **aha152x=IO-addr, IRQ, SCSI-Id, Reconnect, Parity, Debug** sets parameters for Adaptec 152x controller.
- **mcd=IO-addr, IRQ, Mitsumi_Bug_Wait** sets parameters for Mitsumi CD-ROM drive.
- **sound=0xTaaaId** sets parameters for sound boards. The parameters are specified in a single hexadecimal number. The individual digits have the following meanings:
 - **T** type of board, where T can assume the following values:
 1. FM Synth. (YM3812 or OPL3)
 2. Soundblaster 1.0 to 2.0, Soundblaster Pro and 16
 3. Pro Audio Spectrum 16
 4. Gravis UltraSound
 5. MPU-401 UART Midi
 6. SB16 with 16-Bit DMA number

7. SB16 Midi (in MPU-401 emulation)

aaa I/O-addr

i IRQ

d DMA channel (0, 1, 3, 5, 6, or 7)

- **sbpcd=IO-Adresse, Typ** sets parameters for Soundblaster / Panasonic CD-ROM driver.
- **cdu31a=IO-Adresse, IRQ** sets parameters for Sony CDU-31A CD-ROM driver.

6.3 Daemons

error messages
syslog
Since no terminal is assigned to daemons, they normally do not output error messages but send all error messages and other reports to the syslog daemon. To simplify the detection of errors in the configuration of daemons and other programs, configure the syslog daemon next.

syslogd / klogd
As usual, there are several versions of this daemon under Linux. In the following description we describe the `syslogd / klogd` combination by Dr. G. Wettstein, which is more readily configurable compared to the normal BSD `syslogd`.

Syslog daemon

log file, e-mail,
console
The syslog daemon `syslogd` can write a message that it receives from another daemon to a log file, send it by e-mail to certain users, or display it directly on the console. Settings can be made individually to determine where the message is output for each unit that sends syslog a message and for the priority of such a message. These settings are specified in the file `/etc/syslog.conf`.

priority
One solution that usually suffices is to output especially important messages on the console and to collect all other messages according to priority in a log file. The entries necessary in `/etc/syslog.conf` to achieve this are as follows:

```
*.alert /dev/console
*.crit /dev/console
kern.* /dev/console
```

```
*.debug          /var/adm/debug
*.=info;*.=notice /var/adm/messages
*.warn           /var/adm/syslog
```

All entries consist of the specification of a unit and a priority and the target of the messages. An overview of all defined units and priorities can be found on the on-line Manual page for `syslogd` or for the file `syslog.conf`. The extensions in the version by Dr. Wettstein are described on the Manual page `syslogd`. unit and priority

The files in the directory `/var/log` must already exist when the syslog daemon starts. To create an empty file, it is simplest to use the command `touch` with the respective file as the parameter. files must exist

To check whether the settings in the configuration file are correct, terminate the syslog daemon with the `kill` command and then restart it with the option `-d`. This starts the daemon in debug mode; syslog then displays a matrix showing which messages are written to which files or sent to which users. option -d
matrix

Since in the above configuration all messages are appended to files, the administrator should ensure that the log files do not grow out of proportion. We suggest using a script that `cron` executes regularly to move the log files to another directory and then create them anew. Afterwards the syslog daemon must be notified by means of `/etc/syslogd.reload` to reopen its log files. If problems arise later, the old log files can still be referenced. log files

Printer daemon

The general definition and configuration of printers takes place in the file `/etc/printcap`. This file specifies, for example, whether to include a cover sheet with each new print job or whether a form feed should follow a print job. Furthermore, multiple printer queues can be created, each with its own filter program. /etc/printcap
filters

In order to enable a computer to access print servers, which provide printers via the network, the client machine must be entered in the file `/etc/hosts.lpd` of the print server. This file contains the names of all computers that have access to the printer. The following example shows the file `/etc/printcap` of a Linux machine that /etc/hosts.lpd

lp=	designates printer interface (default is /dev/lp)
rm=	name of remote host that is printer server
rp=	name of remote printer queue (default is lp)
sd=	name of local spool directory
if=	name of input filter
of=	name of output filter
mc#	max. number of possible copies of a document
mx#	max. size of a job in blocks; mx# permits jobs of any size
sc	suppresses multiple output of a document
sf	suppresses output of page feed after every print job
sh	suppresses output of a title page before every print job

Important `printcap` options

does not possess its own printer, but has access to a printer on a workstation:

```
#
# /etc/hosts.lpd
#
hades.demo.de
hermes.demo.de
jupiter.demo.de
```

`printcap` The following example shows the file `/etc/printcap` of a Linux machine that does not possess its own printer, but has access to a printer on a workstation named `zeus`:

```
#
# /etc/printcap: configuration of a remote printer (zeus)
#
lp:lp=:rm=zeus.demo.de:sd=/usr/spool/lp:mx#0
```

`queue` Every line of the `printcap` file defines a printer queue. The individual options are delimited by colons, whereby the first attribute specifies the name of the queue (`lp`). If an entry extends beyond a single line, the end of a continued line is marked with a backslash (`\`).

The table above lists the most important options. More detailed information can be found in the corresponding on-line Manual page. Text options end with an equal sign (=), numeric ones with a pound sign (#).

International character sets (e.g., French, Spanish, German) pose a problem in outputting text files to a printer. For German, this includes the umlauts. The installation of a corresponding filter can deliver satisfactory results here. Such a filter can be realized in various ways. The following example for German lists a simple C program that converts umlauts and sends the printer a carriage return (CR) after each line. Alternatively, the UNIX command `tr` for character conversion could serve this purpose.

options

international
character sets

C program

`tr`

```

/*****
* Umlaut conversion for EPSON printer
*****/

#include <stdio.h>

main (int argc, char *argv[])
{
    int ch;

    while ((ch = getchar ()) != EOF)
    {
        /* printer needs CR+LF */
        if (ch == '\n')
            putchar ('\r');

        /* convert ISO to PC */
        switch (ch)
        {
            case 228: /* "a */
                ch = 132;
                break;
            case 246: /* "o */
                ch = 148;
                break;
            case 252: /* "u */
                ch = 129;
                break;
            case 196: /* "A */
                ch = 142;
                break;
            case 214: /* "O */
                ch = 153;
                break;
            case 220: /* "U */
                ch = 154;
                break;
            case 223: /* "sz */
                ch = 225;
                break;
            case 167: /* paragraph */
                ch = 21;
                break;
            default:
                break;
        }
    }
}

```

```
        putchar (ch);  
    }  
}
```

Such a filter is linked via the option `if` (input filter) or `of` (output filter) in the `/etc/printcap` file. An output filter is initialized only once for multiple waiting print jobs; an input filter is restarted for each print job.

```
#  
# /etc/printcap: printer configuration  
#  
lp:lp=/dev/lp1:sf:sd=/usr/spool/lp:mx#0:sh  
  
# text queue  
txt:lp=/dev/lp1:sd=/usr/spool/txt:\  
    if=/usr/spool/lp/epson:mx#0:sh:sf  
  
# PostScript queue  
ps:lp=/dev/lp1:sd=/usr/spool/ps:\  
    if=/usr/spool/lp/Postscript:mx#0:sh:sf
```

Linking a filter in `/etc/printcap`

multiple queues

Registering multiple printer queue s enables switching between filters as needed. The choice of the correct queue is specified in a parameter of the `lpr` command. The following allows printing a text with `umlauts`:

```
linux1:/home/tul> lpr -Ptxt Umlaut.txt
```

PostScript

With the right filter, an ordinary matrix, ink jet, or laser printer can easily be converted to a full-scale PostScript printer. The PostScript interpreter Ghostscript is registered as the filter, which cannot be done directly, but via a shell script:

```
#!/bin/sh  
#  
# PostScript printer filter  
#  
  
DEVICE=epson  
  
exec /usr/bin/gs -q -sPAPERSIZE=a4 -dSAFER\  
    -sDEVICE=$DEVICE -sOutputFile=- -
```

PostScript filter

The above example assumes that the printer is an Epson-compatible device. Another type of printer could be substituted by adjusting the `sDevice` parameter. Ghostscript provides a list of possible options:

```
hermes:/home/uhl> gs -help
Ghostscript version 2.6.1 (5/28/93)
Copyright (C) 1990-1993 Aladdin Enterprises, Menlo Park, CA.
Usage: gs [switches] [file1.ps file2.ps ...]
Available devices:
  x11 dmp bj10e bj200 cdeskjet cdjcolor cdjmono cdj500
  cdj550 declj250 deskjet dfaxhigh dfaxlow djet500 djet500c
epson
  eps9high epsonc escp2 ibmprio jetp3852 laserjet la50 la75
  lbp8 ln03 lj250 ljet2p ljet3 ljet4 ljetplus m8510
  necp6 oki182 paintjet pj pjxl pjxl300 r4081 t4693d2
  t4693d4 t4693d8 tek4696 bmpmono bmp16 bmp256 bmp16m
gifmono
  gif8 pcxmono pcx16 pcx256 tiffg3 pbm pbmraw pgm
  pgmraw ppm ppmraw bit
...
hermes:/home/uhl>
```

The `aps` filter package filter package poses an interesting alternative to creating multiple queues. Here it suffices to register a single filter script, which then automatically recognizes the type of the document to be printed and activates the appropriate filter. Thus you can output any kind of data via a single queue:

`aps-filter`

automatic recognition

```
zeus:/home/uhl> lpr postscript.ps tex.dvi text.txt
```

The `printcap` file for the `aps` filter and an HP Deskjet looks like this:

```
#
# /etc/printcap: Printer configuration for aps filter
#
# apsfilter setup Wed Oct 5 17:24:41 MET 1994
#
# APS_BASEDIR:/usr/local/apsfilter
#
lp|lp2|djet500-a4-auto-mono|djet500 auto mono:\
  :lp=/dev/lp1:\
  :sd=/usr/spool/djet500:\
  :if=/usr/local/apsfilter/filter/aps-djet500-a4-auto-mono:\
  :mx#0:\
  :sh:
ascii|lp1|djet500-a4-ascii-mono|djet500 ascii mono:\
```



```
:lp=/dev/lp1:\n:sd=/usr/spool/djet500:\n\n:if=/usr/local/apsfilter/filter/aps-djet500-a4-ascii-mono:\n:mx#0:\n:sh:\n\nraw|lp3|djet500-a4-raw|djet500 auto raw:\n:lp=/dev/lp1:\n:sd=/usr/spool/djet500:\n:if=/usr/local/apsfilter/filter/aps-djet500-a4-raw:\n:mx#0:\n:sh:
```

6.4 Serial login

To configure a login via a serial interface, instead of the normal `getty` command, use the `mgetty` package by Gert Döring. `mgetty` makes it possible to operate multiple services on a single port simultaneously. In addition to normal login, faxes can be sent and received. An extended version named `vgetty` makes it possible to connect a Zyxel modem as an answering machine.

`mgetty` is normally started by the `init` process during booting. This requires an entry in the file `/etc/inittab`:

```
#\n# Start mgetty on port /dev/ttyS0\n#\nS1:45:respawn:/usr/sbin/mgetty ttyS0
```

6.5 Fax

Sending and receiving faxes can be configured with the `mgetty/sendfax` package.

Receiving

`mgetty` automatically saves an incoming fax in the directory `/var/spool/fax/incoming` as a G3 file, then sends an e-mail message to the user `faxadmin` to report the fax arrival:

```
Date: Wed, 11 Jan 95 14:24 MET\nFrom: Fax Getty <root@hn-net.de>\nTo: faxadmin@hn-net.de\nSubject: fax from " +49 9344 1636"\n\nA fax has arrived:\nSender ID: " +49 9344 1636"
```

```

Pages received: 1

Communication parameters: +FCS:0,3,0,2,0,0,0,0
  Resolution : normal
  Bit Rate : 9600
  Page Width : 1728 pixels
  Page Length: unlimited
  compression: 0 (1d mod Huffman)
  Error Corr.: none
  Scan Time : 0

Reception Time : 0:46

Spooled G3 fax files:

  /usr/spool/fax/incoming/fnf13dbfcS0-+49-9344-1636.01

regards, your modem subsystem.

```

Message from mgetty to faxadmin

If the directory `/usr/local/bin` contains a script named `new_fax`, then it is executed after the successful receipt of a fax. Here other actions, such as conversion to another graphic format or printing, can be realized. The `ppm` tool tools supports converting a G3 file to any other formats. `ppm tools`

```

#!/bin/sh
#
# new_fax: convert incoming g3 faxes to GIF.
#

shift 3

for i in $*
do
    /usr/local/netpbm/g3topbm $i |
    /usr/local/netpbm/ppmtogif > $i.gif
    rm $i
done

```

A script to display all received faxes could look like this:

```

#!/bin/sh
#
# faxview - displays incoming faxes
#
xloadimage -geometry 1000x720+10+10 -xzoom 50
/var/spool/fax/incoming/*.gif

```

Sending

Faxes are sent with `sendfax`. The file to be sent must be in G3 `sendfax` format. A PostScript file can be converted easily with Ghostscript:

```
#!/bin/sh
#
# psfax - Versendet eine PS-Datei als Fax.
#
echo 'Converting PS to G3 fax ...'

gs -q -sPAPERSIZE=a4 -sDEVICE=dfaxhigh
-sOutputFile=/tmp/$2.fax - <$2

echo 'Sending fax to' $1

sendfax $1 /tmp/$2.fax

rm /tmp/$2.fax
```

Script to send PostScript files as faxes

log file In the event of any problems, refer to the log file `/var/spool/fax/Faxlog`, which gives the protocol for all actions of the `sendfax` command.

6.6 Streamers and CD-ROM

Many PC configurations today include a streamer and/or a CD-ROM drive. Linux supports such mass storage media, too. If purchasing a new streamer or CD-ROM drive, choosing a SCSI device simplifies Linux installation. Due to the standardization of the command set for SCSI devices, configuration proves quite easy. If the SCSI driver has already been compiled into the kernel, it suffices to make the appropriate entries in the `/dev` directory via the command `mknod`. Most Linux distributions create these device files automatically during installation. For example, to configure CD-ROM drives (`scd?`) and SCSI streamers (`rmt?`), the following entries must exist:

```
ls scd* rmt*
crw-rw-rw- 1 root  root   9,  0 Jan 23 1993  rmt0
crw-rw-rw- 1 root  root   9,  1 Jan 23 1993  rmt1
brw-rw-rw- 1 root  root  11,  0 Jan 23 1993  scd0
brw-rw-rw- 1 root  root  11,  1 Jan 23 1993  scd1
linux1:/dev>
```

device-specific files

If these entries do not exist, the system administrator can generate them with the following commands:

```
linux1:/dev>mknod /dev/rmt0 c 9 0  
linux1:/dev>mknod /dev/scd0 b 11 0
```

Because of the lack of standards for other drivers, the configuration of a floppy streamer or a CD-ROM drive with its own AT bus controller can prove significantly more problematic. Meanwhile, however, corresponding drivers for the most common models have become available under Linux; in order to be used, these also must be compiled into the kernel.

floppy streamer

Administration

Upon completion of the installation of the Linux system and the most important aspects of configuration, it is time to enable users to access the system. In addition, there will soon be requests for applications that are not contained in the installation packages. After some time new versions of some system components will appear and need to be installed to keep pace with continuing development. These kinds of jobs are collectively termed *system administration*. Since system administration tasks on all UNIX systems are very similar, we refer the reader to the standard literature and here only share some tips and notes to Linux-specific details.

users

upgrades

system administration

7.1 The administrator

Only the superuser `root`, that is, the system administrator, can modify the configuration files of the system. The corresponding access privileges of these files guard them against unauthorized use. The system administrator generally has access to all files and can modify them at will.

root

access

This also means that the administrator could crash or erase the entire system. For example, imagine that an administrator were carelessly to enter the following command to delete all files in a directory (with options set to remove all subdirectories recursively and without confirmation) and do so from the root directory (/):

```
linux1:/> rm -rf *
```

This would immediately delete the entire system. If this command were to be executed without `root` permissions, then the

delete all
permissions

access privileges of the system files and directories would guarantee that at worst all the user's own files would be deleted. However, this would have no influence on other users or the overall system.

caution Thus, a system administrator must proceed with extreme caution and only log in as `root` to modify a system file or to install a new program.

7.2 Booting

booting To facilitate understanding of the system, we next describe how a Linux system boots and which programs and scripts are processed during booting. Independent of the operating system, the *master boot record* (MBR) is loaded first during booting. The partition tables and the loader program that loads the boot sector of the active partition are located here.

MBR

LILO If the Linux Loader (LILO) is installed at the beginning of the active partition, then it starts first and presents a choice of various Linux kernels and DOS. If the user selects a Linux kernel, the respective kernel image is loaded and started. Optionally, after selection of the image, parameters can be passed to the kernel itself or to the initialization process.

parameters

resolution First the kernel initializes the video board and possibly prompts for the desired screen resolution. Then it installs the various device drivers, which usually each display a comment. Then the kernel mounts the root file system and starts the process `init`.

drivers

run levels Like UNIX System V, Linux has various run levels, which are specified in the file `/etc/inittab`. These are various configurations in which only certain system components are activated. Normally the system starts in multi-user operation. This means that multiple `getty` processes are started for the console and optionally for the serial ports. In addition, in this mode all network daemons are activated.

multi-user

getty

single-user mode Single-user mode provides an alternative that is intended primarily for system administration. This mode is activated when the option `single` is specified during booting. This option is not evaluated either by LILO or the kernel, but passed to the `init` process after the kernel starts. Another run level might start a graphic login prompt (`xdm`) instead of the usual terminal login, for example.

Because `init` is the first process that the kernel starts, it is always assigned process number 1 and is the parent of all further processes. The `init` process also executes various scripts in the directory `/etc/rc.d`, which usually begin with `rc`. These scripts reinitialize various system files and mount the local file systems. NFS file systems are not mounted yet, because the network and the respective daemons have not been started. The exact sequence and ordering of the various scripts can vary from system to system. In the Slackware distribution, the following files reside in the directory `/etc/rc.d`:

init

/etc/rc.d

scripts

```
/etc/rc.d> ls
rc.0*      rc.K*      rc.S*      rc.inet1*  rc.local*
rc.6*      rc.M*      rc.font*   rc.inet2*  rc.serial*
```

- **rc.S** — This script is invoked first on booting; it initializes the system.
- **rc.serial** — initializes the serial connections. This script can be invoked optionally from the script `rc.S`.
- **rc.M** — multi-user setup. Here the most important daemons are launched.
- **rc.font** — optionally activates a different font for the console.
- **rc.inet1** — initializes the lower layers of the TCP/IP system. Here the IP address, the host name, and the routing table are set.
- **rc.inet2** — starts the network daemons.
- **rc.0** — invoked on shutdown.
- **rc.K** — invoked on switching from multi-user mode to single-user mode.

The following example shows a slightly cropped version of an `rc.S` script:

```
#!/bin/sh

PATH=/sbin:/usr/sbin:/bin:/usr/bin

# enable swapping
/sbin/swapon -a

# Start update.
/sbin/update &

# Test to see if the root partition is read-only, like it
ought to be.
READWRITE=no
if echo -n >> "Testing filesystem status"; then
```

```
rm -f "Testing filesystem status"
READWRITE=yes
fi

# Check the integrity of all filesystems
if [ ! $READWRITE = yes ]; then
  /sbin/fsck -A -a
  # If there was a failure, drop into single-user mode.
  if [ $? -gt 1 ]; then
    echo "*****"
    echo "fsck returned error code - REBOOT NOW!"
    echo "*****"
    /bin/login
  fi
  # Remount the root filesystem in read-write mode
  echo "Remounting root device with read-write enabled."
  /sbin/mount -w -n -o remount /
  if [ $? -gt 0 ]; then
    echo "Attempt to remount root device as read-write failed!
    This is going to"
    echo "cause serious problems... "
    read junk;
  fi
else
  echo "Testing filesystem status: read-write filesystem"
  if [ -d /DOS/linux/etc -a -d /DOS/linux/dev ]; then # no warn
    for UMSDOS
    cat << EOF

*** ERROR: Root partition has already been mounted read-write.
Cannot check!

For filesystem checking to work properly, your system must
initially mount
the root partition as read only. Please modify your kernel
with 'rdev' so that
it does this.

EOF
echo -n "Press ENTER to continue. "
read junk;
fi
fi

# remove /etc/mtab* so that mount will create it with a root
entry
/bin/rm -f /etc/mtab* /etc/nologin /var/adm/utmp

# Looks like we have to create this.
cat /dev/null >> /var/adm/utmp

# mount file systems in fstab (and create an entry for /)
# but not NFS because TCP/IP is not yet configured
/sbin/mount -avt nonfs

# Configure the system clock.
# This can be changed if your system keeps GMT.
if [ -x /sbin/clock ]; then
  /sbin/clock -s
fi

# Run serial port setup script:
# (CAREFUL! This can make some systems hang if the rc.serial
script isn't
# set up correctly. If this happens, you may have to edit the
file from a
# boot disk)
#
#/bin/sh /etc/rc.d/rc.serial
```

Excerpt from the file /etc/rc.d/rc.S

After `rc.S`, for multi-user mode the script `rc.M` executes. This script starts the most important daemons. If a network is present, the scripts `rc.inet1` and `rc.inet2` are invoked here.

network

7.3 Shutdown

As with all UNIX systems, one should avoid simply turning off a Linux computer. Instead, the system must be shut down properly with the command `shutdown`. The reason for the necessity of the shutdown procedure is that, due to the internal cache of the kernel, usually all data that were written by programs to the hard disk interface have not yet physically been stored on the hard disk. Furthermore, frequently needed information, such as the i-node table and the superblock of the file system, are likewise held in RAM. Turning off the computer without using the command `shutdown` can lead to inconsistencies on the hard disk and resulting data loss.

shutdown

cache

i-node tables

The command `shutdown` ensures that all buffers have been transferred to the storage media and that all processes are terminated properly. The command `sync` writes buffers to the hard disk without shutting down the system. This command is seldom used, however.

shutdown

sync

7.4 The Linux directory tree

To assist new Linux users and inexperienced system administrators in gaining an orientation in the system, this section describes the most important directories of a typical Linux installation. The organization of the Linux file system has been determined in the Linux File System Standard, which resides as a PostScript file on the usual FTP servers along with other documents. This standard has been recognized by most of the producers of distributions and packages, and the following description assumes a typical distribution.

orientation

file system standard

distributions

The directory `/` is the root of a Linux directory tree. It is thus called the root directory. Beyond the Linux kernel image files that are needed for booting and the most important subdirectories, it should

root directory

image files

home directory
/root

/etc/passwd

contain no other files. System administrators frequently use the root directory as their home directory. However, we recommend creating a separate directory for this purpose, e.g., `/root`. This makes it easier to distinguish the administrator's configuration files from system files. To create a new home directory, use an editor to modify the respective entry in the file `/etc/passwd`. (Also observe the warning on page 119.)

Directories in the root directory

- programs
 - `/bin` —The most important programs that need to be available even if `/usr` is inaccessible reside in this directory. These include, e.g., the commands `mv`, `cp`, `cat`, and `rm`. Contrary to `/sbin`, which contains only programs for booting and system administration, the programs in `/bin` are intended for all users. All other commands that are not essential in case of emergency reside in `/usr/bin` (also see `/usr`).
- LILO
 - `/boot` —This directory contains the map files of the Linux Loader and backups of the old boot sector and partition table. These files are normally used only by LILO or created automatically by LILO.
- configuration
 - `/conf` —If this directory exists, it contains exclusively configuration files, which otherwise reside in `/etc` or other directories. In this case, rather than actual files, the directory `/etc` contains only symbolic links to the files in `/conf` or its subdirectories, such as `/conf/net`. However, the simplest installations get by without `/conf`.
- devices
 - `/dev` —As the name `/dev` suggests, this directory contains all device files, which are special files assigned to I/O drivers (also see section 2.6).
- CD drive
 - `/dist` —This is where the Unifix and Linux Universe distributions mount their CD. Packages and programs that were not installed on the hard disk exist on the hard disk as links to files in subdirectories of `/dist`.
- configuration files
 - `/etc` —The `/etc` directory contains configuration files. These include the files `passwd` and `group`, which contain user and group information, respectively, and the configuration files of the TCP/IP daemons such as `services`, `inetd.conf`,

and `exports`. Before the file system standard, this directory frequently contained daemons and system programs such as `init` and `update`; these have now been moved to the directory `/sbin` or `/usr/sbin`.

- **/etc/Isode** —This directory contains configuration files for the Isode package (see section 9.16). Isode
- **/etc/X11** —The file system standard assigns this directory for local X11 configuration files. These include `XF86Config` with general settings for the server and the monitor, `Xmodmap` with the keyboard layout under X11, `xinitrc`, and the files for `xdm`. X11
- **/etc/init.d** —In some distributions this directory contains the actual `rc` scripts used by the system during booting and shutdown. These scripts are invoked by System V systems via symbolic links in the directories `/etc/rc0.d` to `/etc/rc6.d`. rc scripts for booting & shutdown
- **/etc/keytables** —The file system standard provides this directory for the keyboard layout maps, which can be loaded on booting. American distributions sometimes use `/usr/lib/keytables` or `/usr/lib/kbd/keytables`. keyboard layout
- **/etc/ppp** —The PPP configuration files reside in this directory. PPP
- **/etc/rc0.d** to **/etc/rc6.d** —These directories are used by distributions where the startup scripts reside in `/etc/init.d`. The scripts in these directories are executed by `init` on a change of the run level. The directories `/etc/rc?.d` contain only link to files in `/etc/init.d`. scripts run level
- **/etc/rc.d** —Instead of in `/etc/init.d`, the scripts that are invoked on system startup by `init` can also reside directly in `/etc/rc.d` or `/etc`. system startup
- **/etc/skel** —The files in this directory are automatically copied into the home directory of the user on creation of a new user with `useradd -m`. The directory normally contains examples of configuration files, including `.cshrc`, `.bashrc`, `.Xdefaults`, and `.emacs`. new user
- **/FTP** —In some distributions this directory is used by the FTP server daemon. Users logging into the server as `ftp` or `anonymous` can access only subdirectories of `/FTP`. Other distributions use the directory `/home/FTP` for this purpose. ftp or anonymous

- home directories
 - **/home** —This directory encompasses a home directory for each user except `root`. Each such subdirectory holds user-specific configuration files. Aside from these personal files of the user, no programs should be installed here.

Since this directory usually resides on a separate partition, we do not advise putting the home directory of `root` in this directory as well. If this file system should not be mountable due to an error, then even the administrator might not be able to log into the system to correct the error.
- installed packages
 - **/install** —The installation program of some distributions employs this subdirectory to store information on installed packages. Other distributions use special directories under `/var` or `/usr`.
- libraries
 - **/lib** —The images of the most important shared libraries of the system reside in directory `/lib`. These images are the parts of the shared libraries that contain the actual routines. They are loaded with the programs using them. The other parts, called *stubs*, are stored in the directory `/usr/lib`. They are linked to the programs and contain only links to the actual routines. Shared libraries that are not absolutely necessary for booting and administration, such as the libraries of the X Window system, should be stored in another directory under `/usr`. For X11 this is `/usr/X11R6/lib`. `/lib` contains only symbolic links to these libraries.
 - **/local** —In some CD distributions a symbolic link from `/usr/local` points to this directory. Local programs that are not contained on the CD should be installed here.
- local programs
 - **/lost+found** —This directory is created automatically with the creation of a file system of type `ext2` and is used by utilities such as `fsck`.
- ext2
 - **/mnt** —This directory should be empty and is often used to temporarily mount diskettes or file systems of other computers via NFS.
- mounting
 - **/proc** —Normally the `proc` file system is mounted here. The `proc` file system is a special file system that stores information on the kernel and running processes as subdirectories and files. These files can be read as text, thus permitting easy access to this information (also see section 3.2).
- kernel info

- **/root** —Although this directory is optional, most Linux distributions create it. This is the superuser's home directory (`root`). Normally this directory is not on the same partition as the home directories of normal users. superuser's home
- **/sbin** —This directory contains only the most important programs and commands required for booting the system and for basic system administration. These include `getty`, `init`, `update`, `fdisk`, `fsck`, `ifconfig`, `ping`, and `lilo`. Programs that are needed by users other than `root` reside in `/bin` or in `/usr/bin` if they are not absolutely necessary in the event of an emergency. system programs
- **/shlib** —Shared libraries for the iBCS2 emulation are stored here. shared
- **/tftpboot** —Some distributions use this directory for the `tftp` daemon. If so, then access per `tftp` can be restricted to this directory. tftp daemon
- **/tmp** —This directory is used by many programs for temporary files. All users have read/write access to `/tmp`. Files in this directory can normally be deleted when no processes are running. Except for the administrator, no other user should be logged in while these files are being deleted. Often the files in the `/tmp` directory are deleted automatically on system startup by means of an entry in the file `/etc/rc.d/rc.local`. temporary files
- **/user** —This directory, inasmuch as it exists, is normally empty and is used for mounting. mounting
- **/usr** —This directory contains almost all other important subdirectories that are not directly needed for booting the system. Separating machine-specific configuration, essential programs for the system administration, and log or spool files from the programs in `/usr` enables using `/usr` from a CD or for multiple machines from a common NFS server. In this case the `/usr` directory must be mounted write-protected. user programs, libraries, Manual pages, and configuration

The most important programs for system administration and the necessary libraries must reside in the root file system so that in the event of a system error where the CD or the NFS server becomes inaccessible, the error can be corrected. The root file system should be as small as possible to allow shared utilization of as many programs and as much disk space as possible.

log and spool files

- **/var** —This directory includes all files that are written to frequently and whose size changes often. These include especially log and spool files. Many of the subdirectories under **/var** were previously under **/usr**. To be able to mount **/usr** for simultaneous read-only access per NFS, these dynamic subdirectories were transferred to the directory **/var**. They include **/var/spool** with the subdirectories for mail and the printer queues, **/var/adm** with the system log files, and **/var/lock** with the lock files.

Subdirectories under /usr

- X11R5 • **/usr/X386** —This is the start of the directory tree of older versions of X11 packages. As of Release 6 the subdirectory **/usr/X11R6** is used instead.
- X11R6 • **/usr/X11R6** —This is the directory of the X Window system since Release 6. The directories **/usr/lib/X11** and **/usr/bin/X11** are links to this directory tree.
- GNU Ada • **/usr/adainclude** —The include files of the GNU Ada compiler reside here.
- programs & commands • **/usr/adm** —This directory is a link to **/var/adm**.
- **/usr/bin** —Most system programs and UNIX commands for users, as well as those for the administrator that are not absolutely needed in case **/usr** cannot be mounted, reside here. This separation of UNIX commands into those stored in **/bin** or **/sbin** and those stored in **/usr/bin** is not always handled consistently. In case of doubt when looking for a command, it pays to look in both directories. Both **/bin** and **/usr/bin** should always be included in the environment variable for the path (**PATH**).
- X11 • **/usr/bin/X11** — X11 programs are normally installed in this directory. However, this is usually a symbolic link to **/usr/X386/bin** or **/usr/X11R6/bin** in the newer X11 releases. This directory should be contained in the **PATH** environment variable.
- dictionary • **/usr/dict** —This directory originally contained an English dictionary for the **look** command and other programs for spell checking.

- **/usr/doc** — Documentation that is not available as a Manual page or in Info format is stored in this directory. documentation
- **/usr/etc** — This directory should contain configuration files that are shared by multiple machines. Often this is only a symbolic link to the directory `/etc`. configuration
- **/usr/g++-include** — This directory contains header files for the GNU C++ compiler.
- **/usr/games** — Here we have games and other entertainment programs that are of subordinate importance for serious application. games
- **/usr/include** — This is the directory for include files of the C library. This directory contains the subdirectories `sys` and `linux`, whereby `linux` is a link to a subdirectory of `/usr/src/linux`. include files
- **/usr/info** — This directory is used for the GNU Info system. The files in this directory can be viewed in info mode in the Emacs editor or with programs like `tkinfo`; they provide the primary documentation of GNU programs. GNU Info files
- **/usr/lib** — The static libraries for various programming languages and the stubs for shared libraries are stored here. In addition, this directory contains subdirectories that usually contain help and configuration files of other programs. libraries
- **/usr/lib/X11** — Here we find the configuration data, character sets, color tables, and other files of the X Window system. This directory is usually a link to `/usr/X11R6/lib/X11`. Files associated with the local configuration of X servers, such as `XF86Config`, should actually be stored under `/etc/X11` according to the file system standard. However, few distributions abide by this point. X11 configuration
- **/usr/local** — This directory should contain all additional programs that were not contained in the installation package. Usually this directory contains a complete subdirectory tree consisting of `bin`, `lib`, `etc`, `include`, and `man` directories. As a rule, `/usr/local/bin` is included in the path for programs (`PATH`) and `/usr/local/man` in the path for Manual pages (`MANPATH`). additional packages

- Manual pages • **/usr/man** —The Manual pages are located in subdirectories of `/usr/man`.
- OpenWindows/XView • **/usr/openwin** —In the subdirectories of `/usr/openwin` we find programs and data of the XView package from Sun. The libraries and configuration files are usually in the directory `/usr/openwin/lib`. The most interesting of these files are the definition files of the menu for the OpenLook window manager (`olwm` and `olvwm`). The file names all begin with `openwin-menu`. The window manager itself and other programs reside in `/usr/openwin/bin`.
- installation • **/usr/pkg** —Some CD distributions use this directory to enable separate installation of individual program packages. From here the programs can be installed on the hard disk with an installation program.
- system programs • **/usr/sbin** —As in the directory `/sbin`, here we find primarily programs for system administration, although only ones that are not essential for booting the system. In addition, this directory contains network daemons.
- network daemons
- shared programs • **/usr/share** —This directory contains files that are machine-independent. The contents of the directory can then be mounted on various machines with different architectures per NFS. Examples of such files include the Manual pages and the `terminfo` database.
- source code • **/usr/src** —The subdirectories under `/usr/src` contain the source code of system programs. The most important of these subdirectories is `/usr/src/linux`, where the source code of the Linux kernel is located.
- TeX • **/usr/TeX** —The TeX package is installed in this directory. In accordance with the file system standard, TeX data files should reside under `/usr/lib/TeX`.

7.5 Users and groups

- user ID & groups Every user has a unique user ID and belongs to one or more groups. This information is stored in the files `/etc/passwd` and `/etc/group`. To create a new user, these files are usually not
- useradd modified with a text editor. Instead, the program `useradd` is used.

This program collects all the important information via the command line. Then it modifies the files `/etc/passwd` and `/etc/group` as well as the respective *shadow files*, which contain the encrypted user and group passwords and, for security reasons, can only be read by the superuser.

shadow files

The administrator can simplify the task of managing users and groups by entering the `-D` option with `useradd` once to set default values for the group, the longevity of passwords, and the directory for the home directories of users.

default values

Furthermore, user-specific configuration files, such as `.profile`, `.bashrc` and `.openwin-menu` can be stored in the directory `/etc/skel`. When a new user is created, these files are automatically copied into the user's home directory.

`/etc/skel`

If default values have been defined and the correct files have been stored in `/etc/skel`, then a new user can be created by invoking the following:

new user

```
linux1: /> useradd -m username
```

The user ID will automatically be the next free number. Next, a password has to be assigned with `passwd <user name>`, since the new account would otherwise remain disabled.

user ID & password

In addition to `useradd`, there are commands to modify the settings (`usermod`) and to delete a user (`userdel`). `Chsh` sets the login shell and `chfn` the full name of a user. The Manual pages provide more information on `useradd` and other commands.

modifications

The following example shows the specification of default values and the adding of a new user:

```
dirkl:/etc# useradd -D -g 6 -b /home -f 3 -e 999
dirkl:/etc# useradd -m peter
dirkl:/etc# passwd peter
Changing password for peter
Enter the new password (minimum of 5 characters)
Please use a combination of upper and lower case letters and
numbers.
New Password:
Re-enter new password:
dirkl:/etc#
```

`groups` Groups are managed with the commands `groupadd`, `group-`
`usermod` `mod`, and `groupdel`. To assign a user to another group, the command
`usermod` with the option `-G` is used.

Anonymous FTP

`ftp` In addition to `root`, the user `ftp` (`user`) plays a special role. If this
user exists, any user can log in to the computer with the `ftp` command
without needing to have an account. The user simply assumes the
`anonymous` identity of user `ftp` or `anonymous`, and the system prompts for the
`user ftp` user's e-mail address as password. The home directory of user `ftp`
is then used as the root directory, so that a user who gains access in
this way has access only to certain files.

 However, this requires the presence of the subdirectories `dev`,
`bin`, and `usr` with their corresponding files in the home directory
`ftp directories` of user `ftp`. The exact structure of this file tree is explained in the
on-line Manual page for `ftpd`.

7.6 Shells

`/etc/shells` To allow users to change their shells themselves, the file
`/etc/shells` must list every available shell with its path. This is
often forgotten when a shell is installed later.

 If there is no entry for a shell in the file `/etc/shells`, the
`login shell` shell cannot be used by users as a login shell and there are problems
with various TCP/IP programs. The following example shows the
contents of such an `/etc/shells` file.

```
/bin/sh
/bin/bash
/bin/ksh
/bin/tcsh
```

`chsh` The command `chsh` permits users to change their login shells.
The administrator should also use this command in the configuration
of an account. In this case the respective user name must be passed
as a parameter.

```

dirkl:/home/stefan# chsh
Changing the login shell for stefan
Enter the new value, or press return for the default
Login Shell [/bin/sh]: /bin/bash
dirkl:/home/stefan#

```

7.7 User information

The message that is displayed before the login prompt appears is located in the file `/etc/issue`. This file normally contains entries with a greeting message, the name of the computer, and instructions for users.

After a user has logged in, usually the message from the file `/etc/motd` is displayed. For some systems that use a corresponding login program shadow passwords, this can be configured in the file `/etc/login.defs`. The acronym `motd` stands for “message of the day,” and this file should also be used as such.

Some distributions, including Slackware, overwrite the files `/etc/issue` and `/etc/motd` in the startup scripts of the system. To allow the user to customize these files, the commands that overwrite them must be removed from the scripts. In the Slackware distribution this occurs in `/etc/rc.d/rc.S`.

7.8 Backups

The `tar` command affords a relatively simple way to make backups of important files. `tar` is one of the standard UNIX commands available under Linux, but in its enhanced form from FSF.

For example, to back up all data in the directory `/home/stefan` onto diskettes, the `tar` command can be invoked with option `M` (multivolume mode). When one diskette is full, `tar` prompts for the next. The following example shows the creation of a `tar` archive on diskette:

```

dirkl:/root# cd /home/stefan
dirkl:/home/stefan# tar cvfM /dev/fd0 *

```

Subdirectories are automatically included in this archiving. To restore such a backup onto the hard disk, the `tar` command is

invoked again with option `M` (otherwise it would terminate after the first diskette). The following example restores a `tar` archive from diskettes:

```
dirk1:/root# cd /home/stefan
dirk1:/home/stefan# tar xvfM /dev/fd0
```

streamer Larger backups can be made onto a streamer in the same manner. Instead of `/dev/fd0`, the device driver of the streamer (`/dev/rmt0`) is specified.

GNU tar The `M` option is not available on other UNIX systems that do not use the GNU `tar` command. Therefore, for backups that might be transferred to other UNIX machines, this option should be avoided.

fdformat The `tar` command assumes, as does `MTools`, that the diskettes it uses are already low-level formatted. Likewise the command `mformat` writes a DOS file system only onto a formatted diskette. For actual low-level formatting, use the command `fdformat`, described on page 374 of the Reference section.

7.9 File system management

Another responsibility of the system administrator is managing the file system. In normal operation this task is restricted to checking free memory in regular intervals, monitoring the log files, and from time to time deleting the contents of the `/tmp` directories.

crash In the event of a system crash, however, the administrator must carry out a consistency check of the file system. Most distributions execute a consistency check on each booting. Here, during booting of the kernel, the file system is first mounted as read-only and checked. Then it is mounted again in an `rc` script as read/write. If problems arise, the consistency check needs to be carried out manually.

file system check For this purpose the individual Linux file systems provide a special tool named `fsck` (file system check). The administrator must ensure that the file system to be checked is not mounted and that the appropriate check program is used. For the currently most popular **e2fsck** `ext2` file system, the corresponding tool is `e2fsck`. The following example demonstrates a consistency check for an `ext2` file system:

```

hermes:/root# mount
/dev/sda1 on / type ext2 (defaults)
/proc on /proc type proc (rw)
/dev/sda2 on /usr type ext2 (rw)
/dev/sda5 on /var type ext2 (rw)
/dev/sda6 on /www type ext2 (rw)
/dev/sda7 on /FTP type ext2 (rw)
hermes:/root# umount /FTP
hermes:/root# e2fsck /dev/sda7
fsck.ext2 0.5a, 5-Apr-94 for EXT2 FS 0.5, 94/03/10
Pass 1: Checking inodes, blocks, and sizes
Pass 2: Checking directory structure
Pass 3: Checking directory connectivity
Pass 4: Check reference counts.
Pass 5: Checking group summary information.
/dev/sda7: 9094/212160 files, 750177/845401 blocks
hermes:/root# mount -text2 /dev/sda7 /FTP
hermes:/root#

```

`fsck` is invoked with the partition or file system as a parameter. Normally the program displays any inconsistencies on the console. If no error messages appear, the file system has not been corrupted.

inconsistencies

Special options permit automatic error correction. However, `fsck` program authors usually recommend repairing a damaged file system interactively.

automatic correction

7.10 Upgrades

Because, particularly with Linux, new versions of the kernel, the C library, or the C compiler are released almost every month, one of the tasks of the system administrator is to install these updates in the system. This section explains how to incorporate such updates.

new versions

GCC

The newest version of the GNU C compiler (GCC) for Linux can be downloaded from the FTP server `tsx-11.mit.edu`. It usually encompasses several files compressed with `tar` and `gzip` and containing compiled programs (binaries) for Linux.

C compiler

tar files

To replace the old C compiler with the new version, it usually suffices to unpack the `tar` files in the root directory. The older version is overwritten in the process. To save storage space, the directories of the old version under `/usr/lib/gcc-lib` can be deleted.

unpacking

More detailed instructions for updating are included in the corresponding `README` or `RELEASE` files found in the same `tar` archive.

README

Libraries

C library The Linux C library is likewise downloadable along with the C compiler from the FTP server `tsx-11.mit.edu` in the directory `/pub/linux/packages/GCC`. Installation of a new version of the

tar archive library usually requires two tar archives archive whose names contain the strings “image” and “inc,” respectively. One file

header files contains the actual library files, the other the matching header files. In addition, there is a file with `extra` in its name that contains the libraries for debugging and profiling.

directories These archives must be unpacked in the root directory such that the include files are in the directory `/usr/include` and the library files in `/lib` and `/usr/lib`.

```
stef1:/# tar xvfz /home/strobel/image-4.5.26.tar.gz
./lib/
./lib/libc.so.4.5.26
./lib/libm.so.4.5.26
./lib/libc-lite.so.4.5.26
./usr/lib/
./usr/lib/libc.sa
./usr/lib/libcurses.sa
./usr/lib/libtermcap.sa
./usr/lib/libdbm.sa
./usr/lib/crt0.o
./usr/lib/libc.a
./usr/lib/libm.sa
./usr/lib/libm.a
./usr/lib/libtermcap.a
./usr/lib/libcurses.a
./usr/lib/libdbm.a
./usr/lib/libbsd.a
./usr/lib/libieee.a

stef1:/# tar xvfz /home/strobel/extra-4.5.26.tar.gz ./usr/lib/
./usr/lib/libg.a
./usr/lib/libmcheck.a
./usr/lib/gcrt0.o
./usr/lib/libc_p.a
./usr/lib/libgmon.a
stef1:/# tar xvfz /home/strobel/inc-4.5.26.tar.gz
./usr/include/
./usr/include/arpa/
./usr/include/arpa/FTP.h
./usr/include/arpa/inet.h
...
```

major and minor version

The version number of the libraries consists of two parts, a major and a minor version number. The files that are unpacked above to `/lib` with major version 4 and minor version 5.26, for example, would be called `/lib/libc.so.4.5.26`, `/lib/libc-lite.so.4.5.26`, and `/lib/libm.so.4.5.26`. These files are

symbolic link

accessed via symbolic links that contain only the major version in

their names. For the above files there must be symbolic links to `/lib/libc.so.4` and `/lib/libm.so.4` that reference the actual files.

To install a library with a new minor version, these links need to be modified. Here we demonstrate the installation of a new library with the invocation of the program `ldconfig`:

new version

ldconfig

```
stef1:/# ldconfig -nv /lib ldconfig: version 1.4.3
/lib:
  libc-lite.so.4 => libc-lite.so.4.5.26
  libgr.so.1 => libgr.so.1.3
  libXpm.so.3 => libXpm.so.3.3.0
  libvga.so.1 => libvga.so.1.0.11
  libm.so.4 => libm.so.4.5.26
  libc.so.4 => libc.so.4.5.26
stef1:/# cd /lib
stef1:/lib# ls -l libc* libm* -rwxr-xr-x 1 root root
619524 Apr  4 21:32 libc-lite.so.4.5.26*
lrwxrwxrwx 1 root root 14 Apr 27 23:50
libc.so.4 -> libc.so.4.5.26*
-rwxr-xr-x 1 root root 623620 Apr 19 14:05
libc.so.4.5.24*
-rwxr-xr-x 1 root root 623620 Apr  4 21:31
libc.so.4.5.26*
lrwxrwxrwx 1 root root 14 Apr 27 23:50
libm.so.4 -> libm.so.4.5.26*
-rwxr-xr-x 1 root root 107524 Apr 19 14:05
libm.so.4.5.24*
-rwxr-xr-x 1 root root 107524 Apr  4 21:31
libm.so.4.5.26*
stef1:/lib#
```

In the above example a library with major version 4 and minor version 5.24 is replaced with a version with the same major version and minor version 5.26.

We do not recommend modifying the links manually with the command `ln` or first deleting the old link and then trying to create a new link. Since the command `ln` itself uses the C library, if the link is deleted, no new link can be created. Likewise the other commands depend on the C library and can no longer be invoked. To reestablish the link, the system must be rebooted from a boot diskette (e.g., from the distribution package), the root file system of the hard disk must be mounted, and there the link must be recreated. Creating a boot diskette is described on section 5.5.

NOT manually

boot disk

Kernel

The source code of the newest kernel can be downloaded from most Linux FTP servers. However, it first appears on

source code

Finnish FTP server

Finland's `nic.funet.fi`. To install the newest version of the kernel, we recommend first completely deleting the directory `/usr/src/linux` and then unpacking the tar archive with the kernel source code into the directory `/usr/src`. In the process the subdirectory `linux` is recreated.

Then, as described in the section on configuration and compilation of the kernel (see section 6.2), the drivers to be used can be defined with `make config` and the new kernel can be compiled.

7.11 Boot diskettes

Not only novices accidentally delete important files and make the system unbootable. Such files primarily include kernel images that are loaded by LILO, shells, and files in the `/etc` directory. In such a case there are several approaches to repairing the system.

If the system refuses to start even in single-user mode (see section 7.2), the situation requires a boot diskette. If the installation diskettes or CD of a distribution such as Slackware, Universe, LST, or SLS is available, a Linux system can be booted from it. Instead of launching the installation program, mount the root partition of the hard disk and reconstruct or copy the missing files. A diskette or streamer backup of important files is quite helpful at such a time.

If only the boot kernel on the hard disk is corrupted, a functioning kernel from another system can be copied directly onto a diskette. This contains only the kernel and no file system or additional files. You can boot from this diskette. After booting, the kernel will attempt to mount the root file system. The device or partition that the kernel should attempt to mount can be specified with the program `rdev` (see section 5.5).

If important files in the `/etc` directory have been damaged and booting with the root partition proves impossible, then this requires a diskette with its own root file system. The boot kernel can reside either on the same or on a second diskette. The more elegant variant is certainly with a single boot diskette containing both the boot kernel and the root file system. Such a diskette allows booting of a minimum but independent Linux system from which the partitions of the hard disk can be mounted to correct the error.

If a diskette is used as the root file system, it must not be removed from the drive. This proves quite impractical if you need to read other disks. Here a ramdisk comes to the rescue; on booting, the diskette is copied onto it. This permits using a ramdisk as the root file system, and the floppy disk drive is freed for other diskettes.

ramdisk

Creating such a boot/root diskette is not particularly difficult. First you need a formatted diskette. Here you can use the DOS program `format`, the Linux program `fdformat`, or a preformatted disk. Normally a MINIX file system is created on this diskette.

creating a boot/root disk

fdformat

```
zeus:/root# fdformat /dev/fd0H1440
Double-sided, 80 tracks, 18 sec/track. Total capacity 1440
kB.
Formatting ... done
Verifying ... done

zeus:/root# mkfs -t minix /dev/fd0 1440
480 inodes
1440 blocks
Firstdatazone=19 (19)
Zonesize=1024
Maxsize=268966912
```

Then the diskette is mounted and the most important directories are created.

directories

```
zeus:/root# mount /dev/fd0 /mnt
zeus:/root# mkdir /mnt/etc /mnt/bin /mnt/sbin /mnt/boot
zeus:/root# mkdir /mnt/dev /mnt/lib /mnt/root /mnt/mnt
```

The device files must be created in the directory `/dev`. The `cp` command can be used here. Then the most important programs, libraries, and other files are copied and symbolic links are created.

`/dev`

programs and libraries

```
zeus:/root# cp -a /dev/* /mnt/dev/

zeus:/root# cp /bin/bash /bin/cp /bin/cat /bin/lm /mnt/bin/
zeus:/root# cp /bin/loadkeys /bin/ls /bin/mkdir /mnt/bin/
zeus:/root# cp /bin/rmdir /bin/rm /mnt/bin/
zeus:/root# ln -s bash /mnt/bin/sh

zeus:/root# cp /sbin/fdisk /sbin/mke2fs /mnt/sbin/
zeus:/root# cp /sbin/mkswap /sbin/mount /mnt/sbin/
zeus:/root# cp /sbin/reboot /sbin/shutdown /mnt/sbin/
zeus:/root# cp /sbin/umount /sbin/update /mnt/sbin/

zeus:/root# cp /usr/lib/kbd/keytables/gr-lat1.map /mnt/etc/

zeus:/root# cp /boot/boot.b /mnt/boot/
```

```
zeus:/root# cp /lib/ld.so /mnt/lib
zeus:/root# cp /lib/libc.so.4.5.26 /mnt/lib/
zeus:/root# ln -s libc.so.4.5.26 /lib/libc.so.4
```

link The version number of the C library can deviate from the version 4.5.26 employed in the example. What is important is that, as on the hard disk, a symbolic link that contains only the major version number is created pointing to the actual library.

/etc Now the most important files in directory /etc must be created. The following lists these files and their contents:

/mnt/etc/group:

```
root::0:root
bin::1:root,bin,daemon
daemon::2:root,bin,daemon
sys::3:root,bin,adm
adm::4:root,adm,daemon
tty::5:
disk::6:root,adm
lp::7:lp
mem::8:
kmem::9:
wheel::10:root
```

/mnt/etc/passwd:

```
root::0:0:root:/root:/bin/sh
bin:*:1:1:bin:/bin:
daemon:*:2:2:daemon:/sbin:
adm:*:3:4:adm:/var/adm:
^D
```

/mnt/etc/fstab:

```
/dev/ram / minix defaults
```

/mnt/etc/rc:

```
# initial path
PATH=/sbin:/bin

# start update
update &

# create a new utmp
cat /dev/null >> /etc/utmp

# create mtab
mount -av
```

```
# Loading german keytable
loadkeys /etc/gr-lat1.map
```

```
/mnt/etc/profile:
```

```
export PATH=/bin:/sbin
```

To enable booting from this diskette, a kernel must be copied onto it and the Linux Loader must be installed. Here you need the special `lilo` configuration file (`lilo.conf.bd`) listed next. The file itself and the `lilo` program need not be copied onto the diskette. LILLO

The definition of the root file system and the ramdisk in the kernel are handled with the utility `rdev` (also see section 5.5). rdev

```
zeus:/root# cp /vmlinuz /mnt/
zeus:/root# rdev /mnt/vmlinuz /dev/ram
zeus:/root# rdev -r /mnt/vmlinuz 1440
zeus:/root# rdev -R /mnt/vmlinuz 0
```

```
lilo.conf.bd:
```

```
boot      = /dev/fd0
install   = /mnt/boot/boot.b
map       = /mnt/boot/map
compact

image = /mnt/vmlinuz
      root = /dev/fd0
      label = linux
```

The `lilo` program is invoked with the new configuration file. This overwrites the boot sector on the diskette with the Linux Loader, and the diskette is ready for booting. boot sector

```
zeus:/root# lilo -C lilo.conf.bd
Added linux
```

This concludes creation of the boot diskette. With the command, `umount` it can be removed from the Linux directory tree. umount

X Window System

As the propagation of graphical workstations began, there were scarcely any standards for programming graphical user interfaces (GUIs). Most manufacturers provided their own GUIs with their machines. If an application that was intended to exploit the graphical possibilities of the new machines was to run on different platforms, it had to be developed and maintained in multiple variants. Large institutions that worked with systems from various manufacturers particularly felt the brunt of these problems.

GUIs

platforms

This prompted the Athena Project at the Massachusetts Institute of Technology (MIT) to develop a platform-independent, uniform environment for the development of graphical applications. At first the development of the X Window System received financial support only from DEC and IBM.

MIT Athena Project

In January 1987 12 renowned workstation producers joined to form the X Consortium. The goal of this institution was to promote the further development and standardization of the X Window System and to enable commercial utilization. In the same year the X Window System Version 11 Release 1 (X11 R1, or X11) appeared. Contrary to previous versions, Version 11 had outgrown the research stage. Although the new release was no longer compatible to Version 10, it afforded much greater flexibility and performance. Version X11 achieved commercial breakthrough and quickly developed into the standard graphical user interface for UNIX systems from a broad range of manufacturers.

X Consortium

X11 R1

8.1 Features

The X Window System boasts features that distinguish it from conventional graphical user interfaces such as Apple Finder

superior GUI

and MS-Windows. The following subsections describe the most important concepts of this powerful system.

Openness

Contrary to most other GUIs, the X Window System was conceived from the start as an open system. This means that the developers maintained independence from any manufacturer-specific policy and also that the complete source code is available for free.

The X Window System supports comfortable development of portable and hardware-independent software. The programmer need not worry about the underlying hardware platform. The system supports numerous input and output devices. Interfaces for manufacturer-specific extensions allow connection of specialized hardware.

The manufacturer independence and the extreme portability of the X Window System have won it a high level of acceptance from the workstation sector. Today there is hardly a hardware platform, from PC to mainframe, for which the system is not available. This general availability also proves to be an advantage for Linux users. The X Window System server XFree86 that runs under Linux displays extremely high performance on normal PC hardware. In any case, X11 is equal to commercial X servers, and in many areas it even proves superior.

Client/server architecture

Due to its internal structure, the X Window System distinguishes between the X server and the X clients. The server program is responsible for managing local hardware such as monitor, keyboard, and mouse and provides the interface between the user and the individual X applications, which are the X clients.

In general, a workstation runs only one server, which provides any number of clients with input and handles screen output requested by these clients. However, it is quite possible that one server might manage multiple monitors that are connected to a single workstation, which is particularly interesting for CAD systems.

X protocol

The X protocol provides the only connection between the server and the X clients. Due to this standard protocol, the clients could even be running on other computers on the network (see “Network Transparency” below).

network transparency

The reader needs to be clear that the designations X server and X client deviate from the accustomed terminology. Normally, a *server* is a machine that is superior to its clients in terms of hardware and processes data queries or computations of these clients. Under the X Window System this is usually inverted: the server runs the front end (a normal workstation), while the *client*, a more powerful machine, runs the back end. Often this distinction disappears because the server and the client are running on the same machine.

server & client

This distinction between client and server has some consequences for both the programmer and the user. For example, the programmer has to transfer a graphic that the client has in bitmap form over the network to the server before it can be displayed. These interrelationships need to be considered in the development of X11 applications to avoid creating unnecessary network traffic and thus provoking the resulting loss in execution speed.

bitmap

network traffic

Since an X client is relatively loosely coupled to the server, the inexperienced user can encounter some surprises from time to time. For example, if a client does not respond immediately to user input because of high network or CPU load, many users tend to repeat the input. However, since the X server records every event and definitely sends it to the client, the action could be executed repeatedly, which only very rarely satisfies the intentions of the user.

loose coupling

response time

Network transparency

An important feature of the X Window System is its network transparency. Most workstations on the UNIX sector are connected via network to one another. The X Window System provides a mechanism for redirecting the graphical output of an application to any workstation on the network. This feature permits running computation-intensive programs on the most powerful CPU while the input and output are handled by a smaller workstation on the network.

redirecting output

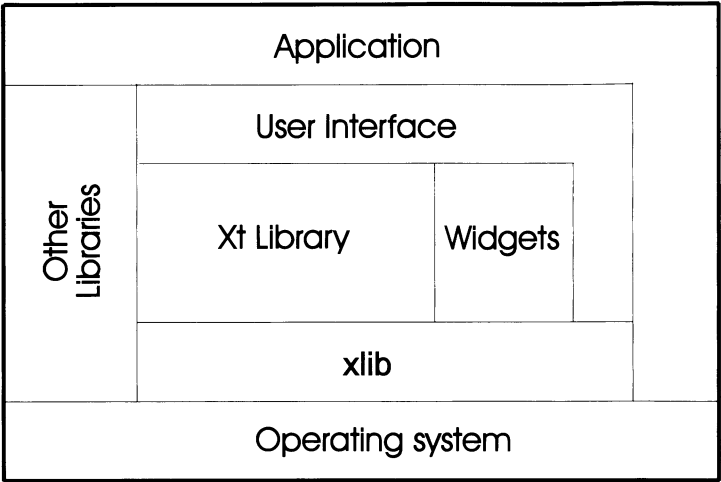


Figure 8.1. Structure of an X11 application

If a commensurately fast network connection is available, the two machines could even be located thousands of miles apart. Although it currently supports only TCP/IP and DECnet, the X Window System is principally not bound to a specific network protocol.

Surprisingly, this network transparency hardly leads to a loss in execution speed compared to conventional graphic systems. The X Window System also supports the display of local applications as well as programs running on remote machines on the network all on one monitor. Therefore the X Window System provides an ideal basis for integration in all possible application domains.

8.2 Structure

The structure of the X Window System reflects several layers (see Figure 8.1). Although the user normally does not perceive these layers, understanding this structure should help to clarify some of the phenomena that the X Window System user experiences.

A very extensive graphics library with a standardized C interface named Xlib provides the basis of the X Window System. Every

invocation of an Xlib routine is translated to a corresponding data stream that is then transferred via the network (X protocol layer) and can be interpreted on another workstation. This also guarantees problem-free communication between workstations of different manufacturers.

protocol layer

X11 does not permit any kind of direct access to the video hardware and does not provide any means for circumventing its standardized interface.

video hardware

Intrinsics

Since Xlib permits only rudimentary graphic operations such as drawing lines and circles and filling surfaces, higher layers were introduced. With the X Toolkit, MIT made a highly mature proposal for the implementation of such libraries. For creating graphical user interface s, the developer normally uses objects on the toolkit level. If an application needs additional primitive graphical output operations, these are realized with direct Xlib invocations.

X Toolkit

The X Toolkit Intrinsics Library (Xt Library) supports the development and use of complex graphical objects such as buttons, text input fields, and selection menus. These objects are collectively called *widgets*.

Xt Library

widgets

Widgets

The actual look and feel of such objects is not determined further by the intrinsics library. A design goal in the development of the X Window System was not to prescribe the look of applications that build on it, but only to provide interfaces for the implementation and use of such widget sets. Thus, over time, various sources produced such widget libraries, sometimes with significant deviations in look and feel. However, this multiplicity seems to have tended to confuse users.

look and feel

Very recent developments show that most manufacturers have agreed on the Motif widget set of the Open Software Foundation as the de facto standard. This suggests that the future will likely see a uniform concept for graphical user interface s under X11.

OSF/Motif

8.3 X resources

flexibility
object-oriented

In their design, the developers of the X Window System created an unusually flexible interface for the configuration of various parameters of the system. The basis was the object-oriented approach of the X Toolkit and the widget set s building on it.

Widget attributes

default settings

Each widget has a number of specific attributes such as position, size, shape, and color. These attributes can be influenced by the programmer as well as later by the user. Every graphical object has internal default settings as determined by the widget developers. The programmer of an X application normally modifies these settings only as much as necessary for the specific requirements.

Resource files

resource manager

When an X application starts, the resource manager loads any new data from the X resource database, whose contents the user can modify in several stages.

8.4 Window managers

window decoration

The window manager is a special client that usually exists only once per workstation. Its job is managing the various windows of a display to allow the user to modify the position and size of windows. The window manager also handles the window decoration, which designates the peripheral regions of a window and its various control elements. Naturally, the respective X client is responsible for the contents of a window.

ICC

The X Window System likewise does not prescribe the look and feel of the window manager. To assure problem-free communication between a given window manager and the clients running under its control, the X Consortium passed the ICC (Inter-Client Communications Conventions).

Users have a broad choice of different window managers, each having its advantages and drawbacks. The standard X distribution, for example, contains the `twm` (Tab Window Manager), which affords

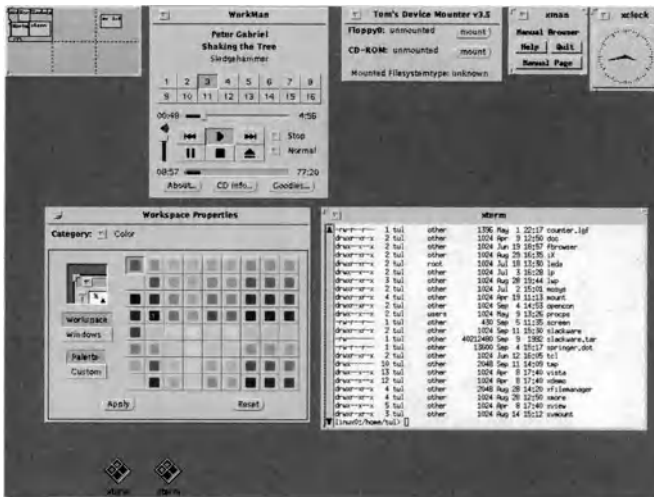


Figure 8.2. The olvwm window manager

little in the way of comfortable operation. The Open Software Foundation (OSF) created the Motif window manager (`mwm`), whose look and feel matches the Motif widget set. Sun Microsystems' OpenLook Window Manager (`olwm`) is available free as source code. `olwm` was extended by a third party to the OpenLook Virtual Window Manager (`olvwm`), which provides almost any number of virtual screen s and allows switching between them with the desktop manager (see Figure 8.2).

OSF Motif
OpenLook

However, most Linux distributions favor `fvwm` by Robert Nation (see Figure 8.3). This window manager is also based on `twm` but consumes significantly less memory than the original. By using a configuration file (`.fvwmrc`) and by setting numerous parameters, the user can exert significant influence on the look and feel of `fvwm`. This allows the user to mimic the look and feel of `mwm` or the window manager by Silicon Graphics. A virtual desktop supports multiple screens, similar to `olvw`.

`fvwm`

virtual desktop

Especially noteworthy is the feature allowing the linking of external modules, which can then communicate with the window manager via a defined interface. One such module is GoodStuff, an icon bar on which the user can place icons for the most frequently

GoodStuff



Figure 8.3. The fvwm window manager

used applications on the desktop and from which they can be launched. Another module permits the coupling of sound s to certain user actions such as opening and closing a window. Another option is the generic window manager (gwm), whose look and feel the user can adapt to either of the two standards (OpenLook or Motif) using a built-in Lisp interpreter.

ctwm is another offspring of twm, but it offers a more pleasing look and feel and is more configurable than its predecessor.

8.5 Toolkits

This section introduces the most important libraries (toolkits) for the production of graphical user interfaces under the X Window System.

Athena widget set

The first available widget set for the X Window System was the Athena widget set, which remains its standard and is included in the MIT distribution. Many early X applications used the Athena widgets. Even today many freeware programs continue to use these widgets.

Unfortunately, the somewhat antiquated look and feel of some of its elements has led manufacturers of commercial applications to adopt more modern toolkits. Due to the similarities between intrinsics-based toolkits, this certainly posed no great problem. However, the look and feel of the Athena widgets has recently received a face lifting. The free availability of Athena's source code made it possible to exchange drawing routines. The modifications led to a Motif-like three-dimensional look. Interestingly, in Linux the new library (`libXaw3D`) can simply replace the old one (`libXaw`).

intrinsics

source code

`libXaw3D`

OpenLook

OpenLook itself represents only a detailed specification of the look and feel for graphical user interfaces. After several years of development, it was released by AT&T and Sun Microsystems with the goal of setting a standard for graphical user interface s under X11.

AT&T, Sun

On the basis of the OpenLook specification, Sun developed a toolkit for the X Window System (`XView`) that is nearly compatible to Sun's previous graphic system, `SunView`. This toolkit was included in Release 4 of the MIT distribution of X11, making it public domain. Unfortunately, `XView` builds directly on `Xlib`, which means that it is not a true widget set. Sun used this toolkit to implement a series of standard applications that are delivered with every Sun workstation (`OpenWindows Deskset`).

`XView`

By delivering OpenLook with its UNIX System V systems, AT&T contributed to OpenLook's wide propagation. As an alternative to `XView`, an X Toolkit-based widget set that likewise met the OpenLook specifications was developed primarily by AT&T.

System V

OSF/Motif

To spur the development of new technologies and standards, especially for UNIX systems, most of the renowned UNIX manufacturers (including Hewlett Packard, IBM, and DEC, but not Sun and AT&T) joined forces in the Open Software Foundation (OSF). This group also recognized the need for a uniform user interface for X11 and initiated the development of OSF/Motif. Especially DEC and HP were involved substantially in the design and implementation.

OSF

Within a few months a user interface emerged that had been adapted to the look of common GUIs on the PC sector.

mwm Its main components were the Motif window manger (mwm) and the X intrinsics-based Motif widget set. In addition, the formal
UIL User Interface Language (UIL) was defined that enabled simple description of Motif-based graphical user interfaces. This description can be translated to a binary format with a special compiler and interpreted with the help of a library. Numerous third parties offer programs for the interactive creation of Motif graphical user interfaces.

de facto standard Due to the large number of companies involved in OSF, Motif soon became the de facto standard of the UNIX world. OpenLook developers Sun Microsystems and AT&T (later transferring to UNIX System Laboratories and Novell) merged with the Motif interface trend in the spring of 1993 with the COSE Initiative (Common
COSE Open Software Environment). This delivered the coup de grâce for OpenLook, at least on the commercial sector. The primary goal of COSE is the standardization of the Motif-based desktop environment CDE (Common Desktop Environment).

license Since OSF did not make the source code of Motif available for free, users have to pay fees for each run-time license. For this reason, Motif was not available for Linux from the start. Meanwhile, however, there are commercial Linux versions of OSF/Motif at reasonable prices. Universities can purchase the source code directly from the OSF and compile it on their workstations, which was done at various sites for Linux systems.

SUIT

University of Virginia SUIT (Simple User Interface Toolkit) was developed at the University of Virginia. Contrary to most X toolkits, this toolkit is also available for other common user interfaces such as MS-Windows and
across platforms Apple Finder. SUIT enables the development of applications that are portable between these operating systems. Its look and feel strongly resembles OSF/Motif, which should help its user acceptance.

interface editor A particular feature of this library is its integrated interface editor, which enables the user to manipulate the interface easily and interactively. For example, the user can modify the position and form

of objects without needing to recompile the application. When the user quits the application, all modifications are automatically written to a file.

InterViews and Fresco

InterViews is an extensive graphical environment that was developed at Stanford University. In contrast to X Toolkit and its widget sets, InterViews has a C++ interface and thus incorporates object-oriented concepts. In addition to its graphics library, the InterViews system provides an interactive interface editor (*ibuild*), a WYSIWYG text editor (*doc*), a C++ class browser (*iclass*), and a powerful, vector-oriented drawing program (*idraw*). Due to the quite substantial overhead, InterViews also puts significant demands on the hardware.

C++ toolkit

Version 6 of X11 included the new toolkit Fresco, which represents a further development of InterViews. Fresco enables embedding graphical objects from other programs, which can be running on arbitrary computers on the network. The interface is specified with a language conforming to CORBA (IDL).

Fresco

CORBA, IDL

XView, Slingshot, and UIT

XView is an independent toolkit that is not based on Xt intrinsics. The Sun Microsystems product was delivered with Release 4 of the X Window System. The programming interface depends strongly on the antiquated SunView, the nonnetwork-transparent, highly kernel-dependent graphical user interface on older Sun workstations. XView implemented the look and feel of OpenLook and, because of the object-oriented approach of its not overly complex structure, was relatively easy to program.

SunView

Xview

To extend the features of XView, a Sun Microsystems employee in England developed the Slingshot extensions. This package offers numerous new graphical objects that harmoniously integrate into the base system. Slingshot extends the object-oriented approach with rudimentary graphical figures such as lines and rectangles. The Slingshot extensions also simplify the programming of the drag-and-drop functionality that was already relatively mature in OpenLook.

Slingshot

To adapt XView (and Slingshot) to the newest developments in the field of programming languages, another Sun employee developed a C++ class hierarchy, UIC, which permits the utilization of all XView objects and simplifies using them. Unfortunately, the integration of functions in derived C++ classes can pose some problems.

8.6 X11 server

A central component of every Linux distribution is the X server from the XFree86 team. XFree86 is a nonprofit organization that is primarily involved in the further development of the X server on Intel-based UNIX systems. XFree86 gives special attention to the support of up-to-date PC graphic boards. For this reason, the XFree86 server also demonstrates performance that sometimes even exceeds what can be expected from RISC workstations. The SVGA server (XF86_SVGA) supports all common VGA graphic boards in all possible resolutions and vertical sweep frequencies. With special servers that exploit the possibilities of modern accelerator boards (Mach, S3), significantly better results can be achieved. However, if the performance of the XFree86 server still does not suffice, or if 24-bit support is required, the commercial Accelerated X by X Inside can be employed.

8.7 Linux as X terminal

A PC running under Linux and XFree86 is superbly suited as an X terminal. This proves especially interesting when expensive software packages are available on a workstation and are intended to be utilized decentrally.

Numerous X server s are available that run under MS-Windows. However, their performance is usually significantly below that of XFree86 servers because they translate the X protocol commands into slower MS-Windows calls.

To give the reader a more precise picture of the possibilities afforded by using Linux as an X terminal, we present an example of such an application in a heterogeneous network. On a network (see

Figure 9.3) we have a powerful graphics program available on a Sun workstation (`sun`) that is to be used by a Linux workstation (`zeus`). First, external access must be permitted to the local X server with the command `xhost`. The permissions can be defined separately for each machine. In general, however, the following input suffices:

xhost

```
zeus:/home/strobel>xhost +
access control disabled, clients can connect from any host
zeus:/home/strobel>
```

Then the Linux user logs in with `telnet` or `rlogin` on the computer `sun`:

telnet

```
zeus:/home/strobel>
zeus:/home/strobel>telnet sun
Trying 141.7.1.20...
Connected to sun.
Escape character is '^]'.

SunOS UNIX (sun)
login: strobel
Password:
Last login: Mon Aug 30 09:59:51 from zeus
SunOS Release 4.1.2 (NEWKERN) #1: Wed Dec 23 11:02:57 MET 1992
sun:/home/strobel>
sun:/home/strobel> setenv DISPLAY zeus:0.0
```

Next the `DISPLAY` environment variable is set so that the X library redirects all graphical output to the Linux machine. In the C shell, as the above example shows, this is done with the command `setenv`, in a Bourne shell with the following:

DISPLAY

```
sun:/home/strobel> export DISPLAY=zeus:0.0
```

If an X client is started now on `sun`, then all windows will appear not on `sun` but on the display of the Linux computer (`zeus`). In this way multiple applications from different workstations can be output on one display and operated from one machine. Linux even spares the user from setting the `DISPLAY` variable, since this is handled automatically by `telnetd`.

Alternatively, an X application can be started on a remote computer with the `rsh` command. This requires a corresponding

rsh

entry in the `.rhosts` file on the respective machine. (more details are provided in section 10.6). The following statement starts the application `xterm` on `sun` and redirects its output to `zeus`. The option `-display` is supported by almost all X applications.

```
zeus:/home/strobel> rsh sun "xterm -display zeus:0.0"
```

To simplify the redirection of output from a remote application, it makes sense to make an entry in one of the pop-up menus of the window manager. How this functions is shown under the configuration of `fvwm` in section 8.9.

8.8 X11 configuration

Installing the X Window System under Linux normally consists of simply unpacking the programs and files from the various `tar` archives. With most Linux packages this occurs during the installation of the operating system and so it poses no problem. The configuration task grows complicated when the X server must be adapted to the available video board and monitor. Then configuration requires modifying the central configuration file `XF86Config`, which resides in the directory `/etc` or `/usr/lib/X11`.

The XF86Config file

This file is divided into sections. Here we list these sections and explain their most important aspects in detail.

- **Files** —defines the paths required by the X server for the RGB color table and the font directories.
- **ServerFlags** —sets the server's general flags, including whether the server can be terminated with <Ctrl-Alt Backspace> and how the server should react to UNIX signals.
- **Keyboard** —defines the connected keyboard and the function of special modifier keys.
- **Pointer** —adapts the mouse driver by specification of its type and the interface used.

- **Monitor** —determines the limit values and the timing data of the monitor(s).
- **Device** —describes video boards.
- **Screen** —assigns monitor, definitions, and video board to an X server.

The section Files

Installation packages usually handle the settings for the RGB table and font paths correctly. Font paths can be entered in individual lines or in a single line delimited by commas. Font server s are specified as transport/hostname:portnumber, for example, tcp/zeus:7100. The following example shows a file section:

```
Section "Files"
RgbPath      "/usr/X11R6/lib/X11/rgb"
# Examples of font server entries:
# FontPath    "tcp/127.0.0.1:7100"
# FontPath    "tcp/font.server.de:7100"
FontPath      "/usr/lib/X11/fonts/misc/"
FontPath      "/usr/lib/X11/fonts/Type1/"
FontPath      "/usr/lib/X11/fonts/Speedo/"
FontPath      "/usr/lib/X11/fonts/75dpi/"
FontPath      "/usr/lib/X11/fonts/100dpi/"
EndSection
```

The section: Server Flags

Two options apply in this section. NoTrapSignal is only interesting for debugging purposes, and DontZap prevents terminating the server with <Ctrl-Alt-Backspace>:

```
Section "ServerFlags"
# If the following option is activated, the X server dumps
# a core file when it receives a signal. Here the option is
# disabled as a comment.
# NoTrapSignals

# Activating the following option disables the key combination
# <Ctrl-Alt-BS> to terminate the servers.
DontZap
EndSection
```

The section: Keyboard

The keyboard driver is adapted here. Standard protocol should always be used. In addition, on non-English keyboards, note that the right

special characters

<Alt> key (**<AltGr>**) must be redefined as `ModeShift` to allow the use of special characters such as “@” and “|.” Many example files employ the American keyboard and use the right **<Alt>** key as `Compose`.

```
Section "Keyboard"
Protocol "Standard"
# Delay and repeat rate for autorepeat
AutoRepeat 500 5
# Numlock to be handled by the server
serverNumLock
# Which LEDs can the user influence (e.g., with xset)
# Xleds 1 2 3
# Function of modifier keys
LeftAlt Meta
RightAlt ModeShift
RightCtl Compose
# ScrollLock ModeLock
# Switching consoles with SysReq (normally not used in Linux)
# VTSysReq
# Command to be executed on opening of the virtuellen terminal
# VTinit "command"
EndSection
```

The section: Pointer

mouse

For the mouse driver, it normally suffices to specify the type of mouse and the interface used. Usually, installation creates a link for this interface as `/dev/mouse`, which can be used here.

```
Section "Pointer"

# One of the following mouse protocols must be selected.
# This need not correspond to the name of the mouse.
# Most generic mice and many Logitech mice
# use Microsoft protocol:
# Protocol "Microsoft"
# All normal bus mice:
# Protocol "BusMouse"
# Many new serial Logitech mice use the following
# (also see ChordMiddle):
# Protocol "Mouseman"
# Older Logitech mice:
# Protocol "Logitech"
# Other mice:
# Protocol "MouseSystems"
# Protocol "MMSeries"
# Protocol "PS/2"
# Protocol "MMHitTab"
# The following should not be used under Linux:
# Protocol "Xqueue"
# Protocol "OSMouse"
Protocol "Microsoft"

# Mouse interface:
Device "/dev/mouse"

# BaudRate and SampleRate (only for some Logitech mice)
# BaudRate 9600
# SampleRate 150
```

```
# Emulation of a 3-button mouse, where 3rd mouse key is
# is simulated by simultaneously pressing left and right
# mouse key.
#   Emulate3Buttons

# ChordMiddle is an option for some 3-button Logitech
# and Mouseman mice.
#   ChordMiddle

# The following resets the DTR line of serial mouse port to 0.
# This option is needed for some MouseSystems mice.
#   ClearDTR
# Some mice also require ClearRTS option, which sets RTS to 0.
#   ClearRTS

EndSection
```

The section Monitor

This section has several purposes. It sets the limit values and the timing data of a monitor, and there can be multiple occurrences of the section. Thereby each monitor is assigned an identifier with which it can later be referenced. The limit values are the maximum horizontal synchronization, the vertical refresh rate, and the bandwidth.

bandwidth

These data can be found in the technical documentation of the monitor. If nothing else is specified, the bandwidth is assumed in MHz, the horizontal synchronization in KHz, and the vertical refresh rate in Hz. Specifying these values in the configuration serves to protect the monitor. On startup, the server tests whether a specified video mode exceeds the monitor's range and discards the mode if it does.

protection

After the technical data of a monitor, the section lists various video modes that are adapted to the respective monitor. The definition of video modes is discussed in detail below.

video modes

```
Section "Monitor"
Identifier "VESA Generic Monitor"
VendorName "Unknown"
    ModelName "Unknown"
    BandWidth 300
    HorizSync 23-38
VertRefresh 50-60
# 640x480@60Hz Non-Interlaced mode
# Horizontal Sync = 31.5kHz
Modeline "640x480" 25 640 664 760 800 480 491 493 525
# 640x480@64Hz Non-Interlaced mode
# Horizontal Sync = 33.7kHz
#Modeline "640x480" 28 640 664 704 832 480 489 492 525
# VESA 640x480@72Hz Non-Interlaced mode
# Horizontal Sync = 37.9kHz
#Modeline "640x480" 31.5 640 664 704 832 480 489 492 520
# VESA 800x600@56Hz Non-Interlaced mode
# Horizontal Sync = 35.1kHz
Modeline "800x600" 36 800 824 896 1024 600 601 603 625
```

```
# VESA 800x600@60Hz Non-Interlaced mode
# Horizontal Sync = 37.9kHz
ModeLine "800x600" 40 800 840 968 1056 600 601 605 628
# VESA 800x600@72Hz Non-Interlaced mode
# Horizontal Sync = 48kHz
#ModeLine "800x600" 50 800 856 976 1040 600 637 643 666
# VESA 1024x768@60Hz Non-Interlaced mode
# Horizontal Sync = 48.4kHz
#ModeLine "1024x768" 65 1024 1032 1176 1344 768 771 777 806
# 1024x768@42.6Hz, Interlaced mode
# Horizontal Sync = 34.8kHz
ModeLine "1024x768" 44 1024 1040 1216 1264 768 777 785 817
Interlace
# 1024x768@43.5Hz, Interlaced mode (8514/A standard)
# Horizontal Sync = 35.5kHz
ModeLine "1024x768" 45 1024 1040 1216 1264 768 777 785 817
Interlace
# VESA 1024x768@70Hz Non-Interlaced mode
# Horizontal Sync=56.5kHz
#ModeLine "1024x768" 75 1024 1048 1184 1328 768 771 777 806
# 1024x768@76Hz Non-Interlaced mode
# Horizontal Sync=62.5kHz
#ModeLine "1024x768" 85 1024 1032 1152 1360 768 784 787 823
# 1152x900@60.14Hz, Non-Interlaced mode
# Horizontal Sync=57.4kHz
##ModeLine "1152x900" 85 1152 1192 1384 1480 900 905 923 955
# 1152x900@48.5Hz, Interlaced mode
# Horizontal Sync=45.6kHz
##ModeLine "1152x900" 62 1152 1184 1288 1360 900 898 929 939
Interlace
# 1152x900@48.5Hz, Non-Interlaced mode
# Horizontal Sync=76.1kHz
#ModeLine "1152x900" 110 1152 1284 1416 1536 900 902 905 941
# 1280x1024@44Hz, Interlaced mode
# Horizontal Sync=51kHz
##ModeLine "1280x1024" 80 1280 1296 1512 1568 1024 1025 1037
1165 Interlace
# 1280x1024@61Hz, Non-Interlaced mode
# Horizontal Sync=64.25kHz
##ModeLine "1280x1024" 110 1280 1328 1512 1712 1024 1025 1028
1054
# 1280x1024@70Hz, Non-Interlaced mode
# Horizontal Sync=74.4kHz
#ModeLine "1280x1024" 125 1280 1296 1552 1680 1024 1024 1032
1062
# 1280x1024@74Hz, Non-Interlaced mode
# Horizontal Sync=78.85kHz
#ModeLine "1280x1024" 135 1280 1312 1456 1712 1024 1027 1030
1064
EndSection

Section "Monitor"
Identifier "EIZO FlexScan T660"
VendorName "EIZO"
ModelName "FlexScan T660i-T/TCO"
    BandWidth 135.0
    HorizSync 30.0-82.0
VertRefresh 55.0-90.0
ModeLine "1024x768" 80 1024 1088 1152 1280 768 770 772 778
ModeLine "1280x1024" 135 1280 1328 1408 1688 1024 1025 1026
1060
ModeLine "1536x1152" 168 1536 1616 1760 2048 1152 1154 1158
1188
EndSection
```

The section Device

video board This section specifies the available video boards. Like Monitor, it can occur repeatedly. For many boards, neither the chipset nor the

dot clocks need to be specified, since the server can collect this data automatic ally on startup. For more complicated boards, however, specification becomes necessary.

```

Section "Device"
Identifier "Generic VGA 16 Color"
#server "XF86_VGA16"
VendorName "GENERIC"
BoardName "GENERIC"
EndSection

Section "Device"
Identifier "Generic SVGA"
#server "XF86_SVGA"
VendorName "GENERIC"
BoardName "GENERIC"
VideoRam 1024
EndSection

Section "Device"
Identifier "Generic SVGA, VideoRam limited to 1MB"
#server "XF86_SVGA"
VendorName "GENERIC"
BoardName "GENERIC"
VideoRam 1024
EndSection

Section "Device"
Identifier "Sigma Legend ET-4000"
#server "XF86_SVGA"
VendorName "Sigma"
BoardName "Sigma Legend ET-4000"
Option "legend"
EndSection

#From: koenig@tat.physik.uni-tuebingen.de (Harald Koenig)
#Date: Sun, 25 Sep 1994 18:55:42 +0100 (MET)

Section "Device"
Identifier "Miro 10SD GENDAC"
#server "XF86_S3"
VendorName "MIRO"
BoardName "10SD GENDAC"
# Clocks 25.255 28.311 31.500 0 40.025 64.982 74.844
# Clocks 25.255 28.311 31.500 36.093 40.025 64.982 74.844
ClockChip "s3gendac"
RamDac "s3gendac"
EndSection

```

Even if the server can determine the clock frequencies clocks automatically, sometimes it still makes sense to specify them manually because the detected values are used as an identifier. The definitions of video modes refer to this identifier in the mode lines of the monitor definition. If there should be fluctuations during the detection of the clock frequency and a clock were identified at 49.5 Hz instead of 50, then the X server might not identify the frequency used by a video mode and would thus terminate with an error message.

computing clocks In addition, the server's automatic detection of clock rates can cause problems with some hardware. Specification of the clock frequencies in a board's definition suppresses automatic detection. To establish the available clock frequencies, remove the `clocks` line from the `Xconfig` file and restart the X server with the option `-probeonly`. The X server then displays the detected clock frequencies and other driver information in text mode and then quits.

The section Screen

screen Each special server (SuperVGA server, monochrome server, S3 server, etc.) can be assigned in this section to a monitor and a video board. When the server starts, it selects the appropriate screen and thus has the data of the video board and the monitor. In addition, this section lists possible video modes that refer to mode lines from the corresponding Monitor section. At run time the user can switch between these modes with the key combinations `<Ctrl-Alt +>` and `<Ctrl-Alt ->` on the numeric keyboard.

```
Section "Screen"
Driver "vga256"
Device "Generic SVGA"
Monitor "IDEK VisionMaster 17 (1)"
Subsection "Display"
Modes "1280x1024" "1024x768" "800x600" "640x480"
EndSubsection
EndSection

Section "Screen"
Driver "accel"
Device "Miro 20SD"
Monitor "IDEK VisionMaster 17 (1)"
Subsection "Display"
Modes "1024x768" "800x600" "640x480"
EndSubsection
EndSection
```

Setting the video modes

synchronization The most difficult and dangerous part of configuration is setting video modes, since this directly defines the synchronization frequencies (timing s) of the monitor, and a low-priced monitor, without protective circuitry against exceeding its frequency limits, can be damaged by incorrect values. To prevent damage, the frequency limits of the monitor should be entered in the configuration file from the start.

The advantage of this kind of configuration of the video mode is that the available monitor can be used to full advantage. For example, a 14" monitor whose maximum horizontal synchronization frequency (HSF) is too low to display 800×600 pixels flicker-free could be operated at a resolution of 800×550 with 72 Hz screen refresh rate (RR).

optimization

no flicker

For each mode, the configuration file specifies the clock to be used as well as four values each for horizontal and vertical synchronization. The specification can be in a single line (as ModeLine) or distributed across several lines. The two definitions in the following example are identical:

vertical timing

```
#           Mode Clock horizontal          vertical
ModeLine "800x600" 45 800 840 1030 1184 600 600 606
624

# The same mode in a different notation:
Mode "800x600"
    DotClock      45
    HTimings      800 840 1030 1184
    VTimings      600 600 606 624
EndMode
```

As an option, flag *s* can be specified at the end of a mode definition, including *interlace*, *+hsync*, *+vsync*, and *csync*. These flags influence the interlace mode and the type of synchronization.

interlace mode

```
Mode "1024x768i"
    DotClock      45
    HTimings      1024 1048 1208 1264
    VTimings      768 776 784 817
    Flags         "Interlace"
EndMode
```

The values for horizontal timing, in order, are: The respective meanings of the values in each group of four numbers are

- the maximum number of pixel *s* after which the picture ceases to be displayed;
- the number of dot-clock ticks to the start of the horizontal synchronization pulse (*sync*), whereby the values are counted ongoing;

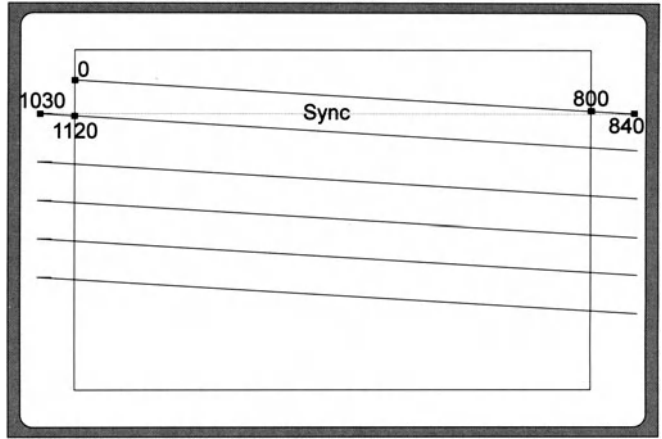


Figure 8.4. Schematic representation of the composition of a screen

- the number of dot-clock ticks until the horizontal synchronization pulse ends and the second guard time of the electron beam begins;
- the total number of dot-clock ticks to the end of a cycle (frame).

```
ModeLine "800x5" 45 800 840 1030 1120 540 540 546 558
```

This example defines a mode with a resolution of 800x540 pixels, which is assigned the name "800x5."

The horizontal resolution is 800 pixels, and after the end of a visible line a guard time begins for the electron beam to rest. The guard time lasts until 840 pixels; then the synchronization pulse begins. It lasts 190 dot-clock ticks until 1030. Then there is a guard time until 1120. Thereupon the next horizontal cycle begins. After 540 horizontal cycles (that is, scan lines), the vertical synchronization intervenes, lasting 6 horizontal cycles. Then there is another guard time until the 558th cycle. After the guard time, the next screen starts.

The files `video.tutorial` and `VideoModes.doc` in the directory `/usr/lib/X11/doc` describe in depth the rules for determining the exact values for such a video mode.

determination of values

However, it is often simpler to find a corresponding entry in one of the many example files and to modify it. There is also a table for the simple spreadsheet program `sc`, which simplifies the computation of the values. This can be found on the FTP server `sunsite.unc.edu` and its mirror servers in the directory `/pub/Linux/X11/install` with the file name `modegen.taz`.

spreadsheet

The limiting factor for simple monitors is usually the maximum horizontal synchronization frequency. This is the frequency with which the electron beam moves from left to right and from scan line to scan line. This frequency is computed by dividing the driving clock rate, specified in MHz, by the largest (right) number of the block for horizontal timing:

horizontal timing

computation

$$f_{horizontal} = \frac{f_{pixel}}{N_{pixel}}$$

In the above example, the required horizontal sweep frequency would be 45 MHz / 1120, or approximately 40 KHz. This is the upper limit for the monitor in our example.

Refresh rate

To compute the vertical synchronization frequency (vertical timing or screen refresh rate), divide the horizontal synchronization frequency by the number of scan lines (i.e., horizontal cycles) necessary for a complete screen. This is the rightmost number in the block for vertical timing.

computation

$$f_{vertical} = \frac{f_{horizontal}}{N_{scan\ lines}}$$

In our example we have 40 KHz/558, or 72 Hz. If 600 lines rather than 540 are to be displayed, then the vertical synchronization frequency falls significantly below 72 Hz, which the user notices as light flickering of the monitor.

flicker

With a better monitor whose maximum horizontal synchronization frequency is, for example, 60 KHz, and a newer video board that

better performance

offers a faster clock, the driving clock frequency could be raised to achieve a higher refresh rate.

copying a mode

To modify an existing video mode, we recommend copying and modifying the mode repeatedly and entering each modified mode in a mode line of the Screen section with a different name. Then start the X server and compare the effects of the modifications by switching modes with **<Ctrl-Alt>** and the + or \$-\$ key on the numerical keypad. If the monitor no longer synchronizes with a new mode, i.e., it fails to show a stable picture, quickly change modes or end the X server with **<Ctrl-Alt-Backspace>** to avoid damage to the monitor.

avoid damage

vgaset

The program `vgaset` provides a valuable aid in adjusting the picture. Started in an `xterm`, it permits interactive manipulation of the picture position. At the touch of a key the borders can be increased or decreased, and the duration of the synchronization signal can be changed. The eight values to be entered for the current settings in the file `Xconfig` are constantly displayed.

Keyboard layout configuration

xmodmap

The X Window System manages the keyboard independently of the kernel. An American keyboard is initialized as the default. Country-specific keyboard tables can be loaded with the utility `xmodmap`. When the X server is launched in a normal configuration, `xmodmap` is invoked with the file `.Xmodmap`. The utility seeks this file first in the user's home directory and then in the directory `/usr/lib/X11/xinit`.

If a given keyboard layout is to be modified systemwide, it must be adapted in the directory `/usr/lib/X11/xinit`. Ready `.Xmodmap` files are included in some distributions, or they can be drawn from FTP servers such as `sunsite.unc.edu` in the directory `/pub/Linux/X11/misc`.

xinitrc

Since `xmodmap` is normally invoked in an `xinitrc` script, the invocation might need to be modified to seek the file `.Xmodmap` in another directory or under another name. In case of doubt, examine the script that starts the X server, usually `startx`.

xkeycaps

Using the `xkeycaps` utility certainly proves to be a more comfortable alternative. It is included in many Linux distributions and can be found on the usual FTP servers. This program offers an



Figure 8.5. xkeycaps

X11 user interface (see Figure 8.5). The user can display the current keyboard layout and interactively change it with the mouse.

American Linux distributions frequently lack other country-specific keyboard layouts. Hence we give an example of a German .Xmodmap as an example of country-specific keyboard adaptation. Note the keyboard symbol number 12, which is often erroneously configured with the symbol `paragraph` instead of `section`.

`paragraph`

```
keycode 8 =
keycode 9 = Escape
keycode 10 = 1 exclam
keycode 11 = 2 quotedbl twosuperior
keycode 12 = 3 section threesuperior
keycode 13 = 4 dollar
keycode 14 = 5 percent
keycode 15 = 6 ampersand
keycode 16 = 7 slash braceleft
keycode 17 = 8 parenleft bracketleft
keycode 18 = 9 parenright bracketright
keycode 19 = 0 equal braceright
keycode 20 = ssharp question backslash
keycode 21 = apostrophe grave
keycode 22 = BackSpace
keycode 23 = Tab
keycode 24 = q Q at
keycode 25 = W
keycode 26 = E
keycode 27 = R
keycode 28 = T
keycode 29 = Z
keycode 30 = U
keycode 32 = O
keycode 33 = P
keycode 34 = Udiaeresis
keycode 35 = plus asterisk asciitilde
keycode 36 = Return
keycode 37 = Control_L
keycode 38 = A
keycode 39 = S
keycode 40 = D
keycode 41 = F
keycode 42 = G
keycode 43 = H
keycode 44 = J
keycode 45 = k K Arabic_kaf
```

```

keycode 46 = l L Arabic_lam Greek_lambda
keycode 47 = Odiaeresis
keycode 48 = Adiaeresis
keycode 49 = asciicircum degree
keycode 50 = Shift_L
keycode 51 = numbersign apostrophe
keycode 52 = Y
keycode 53 = X
keycode 54 = C
keycode 55 = V
keycode 56 = B
keycode 57 = N
keycode 58 = m M mu
keycode 59 = comma semicolon
keycode 60 = period colon
keycode 61 = minus underscore
keycode 62 = Shift_R
keycode 63 = KP_Multiply
keycode 64 = Alt_L
keycode 65 = spaCe
keycode 66 = Caps_Lock
keycode 67 = F1
keycode 68 = F2
keycode 69 = F3
keycode 70 = F4
keycode 71 = F5
keycode 72 = F6
keycode 73 = F7
keycode 74 = F8
keycode 75 = F9
keycode 76 = F10
keycode 77 = Num_Lock
keycode 78 = ScRroll_Lock
keycode 79 = Home KP_7 KP_7 Home
keycode 80 = Up KP_8 KP_8 Up
keycode 81 = Prior KP_9 KP_9 Prior
keycode 82 = KP_Subtract
keycode 83 = Left KP_4 KP_4 Left
keycode 84 = Begin KP_5 KP_5 Begin
keycode 85 = Right KP_6 KP_6 Right
keycode 86 = KP_Add
keycode 87 = End KP_1 KP_1 End
keycode 88 = Down KP_2 KP_2 Down
keycode 89 = Next KP_3 KP_3 Next
keycode 90 = Insert KP_0 KP_0 Insert
keycode 91 = Delete KP_Decimal KP_Decimal Delete
keycode 92 = 0x1007ff00
keycode 93 =
keycode 94 = less greater bar
keycode 95 = F11
keycode 96 = F12
keycode 97 = Home
keycode 98 = Up
keycode 99 = Prior
keycode 100 = Left
keycode 101 = Begin
keycode 102 = Right
keycode 103 = End
keycode 104 = Down
keycode 105 = Next
keycode 106 = Insert
keycode 107 = Delete
keycode 108 = KP_Enter
keycode 109 = Control_R
keycode 110 = Pause
keycode 111 = Print
keycode 112 = KP_Divide
keycode 113 = Mode_switch
keycode 114 = Break

```

8.9 Configuration of X applications

Most X clients come with an *application defaults* file that is copied into the X11 system region (`/usr/lib/X11/app-defaults`). This file contains the important default settings for the application, such as size, position, and color of graphical objects and error messages in the respective country-specific language.

application defaults

Every application was assigned a class name by its programmer that corresponds to the name of its resource file. Class names always begin with a capital letter. Change the background color of `xterm` (class name `XTerm`) in the file `/usr/lib/X11/app-defaults/XTerm`.

class names

Various environment variables (`XFILESEARCHPATH`, `XAPPLERESDIR`) affect the search path for resource files. `XFILESEARCHPATH` allows specification of multiple search paths delimited by colons and handles several special characters as follows:

`XFILESEARCHPATH`

<code>%C</code>	value of customization resource (<code>*.customization</code>)
<code>%L</code>	language, local codeset
<code>%l</code>	language
<code>%N</code>	class name
<code>%T</code>	file type (<code>app-defaults</code>)

A useful definition of this variable could be as follows:

```
XFILESEARCHPATH=/usr/lib/X11/%T:/usr/local/%T/%N:$HOME/%T/%N
```

Resource files will now be searched for in three directories:

1. `/usr/lib/X11/app-defaults/<Class>`
2. `/usr/local/app-defaults/<Class>`
3. `<Home-directory>/app-defaults/<Class>`

Another way to configure X11 applications is the command `xrdb`, which loads the passed resource file into one of the properties (`RESOURCE_MANAGER` or `SCREEN_RESOURCES`) of the X server.

`xrdb`

property A *property* is a global memory region in the X server which can be assigned a name. On launching of an X application, the resource manager evaluates the resource definitions contained in these properties. Configuring applications by means of resource properties particularly makes sense if applications are started on another computer and the local look and feel is to be affected.`xrdb` has a number of parameters:

<code>-all</code>	Operation refers to both properties
<code>-screen</code>	Operation refers only to the property <code>SCREEN_RESOURCES</code>
<code>-global</code>	Operation refers only to the property <code>RESOURCE_MANAGER</code>
<code>query</code>	Displays current contents of a property
<code>merge <file></code>	Merges contents of a file with a property
<code>edit <file></code>	Saves contents of a property to a file
<code>remove</code>	Removes a complete property
<code>load <file></code>	Overwrites a property with contents of a file

`xrdb` commands

`.Xdefaults` In addition, users can create their own `.Xdefaults` files in their home directories and thus override the global default settings.

widget attribute The following list gives an overview of the files and paths that are processed sequentially to determine the current widget attributes on starting an application. If a resource is assigned a value at more than one location, then only the last defined value applies.

- Within an application:
 1. Fallback resources
- Application-specific:
 1. `/usr/lib/$LANG/app-defaults/<class>`
 2. `/usr/lib/X11/app-defaults/<class>`
- New search path:
 1. `$XFILESEARCHPATH`
- User-specific:
 1. `$XUSERFILESEARCHPATH`

2. `$XAPPLRESDIR/$LANG/<class>`
 3. `$XAPPLRESDIR/<class>`
 4. `$HOME/$LANG/<class>`
 5. `$HOME/<class>`
- **Screen-specific:**
 1. `SCREEN_RESOURCES` property (xrdp)
 - **Display-specific:**
 1. `RESOURCE_MANAGER` property (xrdp)
 2. `$HOME/.Xdefaults` file
 - **Host-specific:**
 1. `$XENVIRONMENT` variables
 2. `$HOME/.Xdefaults-<hostname>`
 - **Command line:**
 1. Command-line options

Widget attributes

These resource values are represented in ASCII format. To distinguish them within a resource file, the programmer assigns each application a name (class), which seldom corresponds to the program file (instance). Likewise each widget and widget attribute that is externally configurable has a name and belongs to a class. To uniquely reference a widget, the name does not suffice. Instead, as with a file system, this requires a path that represents an excerpt from the widget hierarchy.

To allow manipulation of the attributes of multiple widgets simultaneously, wildcards (?, *) are permitted in the path. The following is the exact syntax of the resource specification:

```
object.subobject[.subobject...].attribute: value
```

The individual elements have the following meanings:

object	class or name of program
subobject	class or name of widget
attribute	resource name
value	value

.	delimiter
*	wildcard, any number of, or no, names
?	wildcard, any individual name

attribute

The first column of the resource file specifies the attribute to be manipulated. This usually corresponds to a widget resource. However, the programmer can define new, application-specific resources. The hierarchy and the names of the available resources can be taken from the respective Manual page.

```
XTerm*background:      gray90
XTerm*ScrollBar:       true
XTerm*Foreground:      white
XTerm*Background:      gray20
XTerm*IconName:        XTerm
XTerm*WaitForMap:      true
XTerm*MarginBell:      false
XTerm*JumpScroll:      true
```

Excerpt from a resource file

Likewise, individual widget attributes can be collected into classes. Making use of class identifiers can make a resource file significantly shorter and more readable. The attributes `cursorColor` and `pointerColor` of `xterms` both belong to the class `Foreground`. Therefore the following:

```
XTerm*foreground:       green
XTerm*cursorColor:     green
XTerm*pointerColor:    green
```

can be abbreviated as:

```
xterm*Foreground:      green
```

Releases 5 and 6 of the X Window System contain an interactive resource manager (`editres`) that permits comfortable manipulation of all resource values of a running program and saves them on demand to an ASCII file. It is particularly noteworthy that this can be done at run time. Thus the user gets immediate feedback on the effects of

changes. Unfortunately, the `editres` protocol is not yet supported by all widget sets, which naturally restricts the use of the tool. The generated ASCII files can easily be integrated into available resource files or appended to the `.Xdefaults` file.

protocol

Configuration of the window manager

The user can configure not just the look of individual applications, but of most X window managers as well. Since many Linux users probably use `fvwm`, we go into detail only on this window manager. The parameters of this window manager are set in the file `system.fvwmrc` in the directory `/usr/lib/X11/fvwm`. Alternatively, each user can provide a file named `.fvwmrc` in the user's home directory.

fvwm

Using the M4 preprocessor provides additional flexibility. This allows linking additional configuration files or testing conditions, for example. The main file (`system.fvwmrc`) thus becomes relatively comprehensible (see Figure 8.8).

M4 preprocessor

```
#####
#
# system.fvwmrc - fvwm configuration
#
#####
# Paths

ModulePath /usr/lib/X11/fvwm/modules
PixmapPath /usr/lib/X11/pixmaps:/usr/local/lib/pixmaps
IconPath /usr/include/X11/bitmaps/

#####
# External configuration files

include(/usr/lib/X11/fvwm/fvwm.options)

include(/usr/lib/X11/fvwm/fvwm.menus)

include(/usr/lib/X11/fvwm/fvwm.functions)

include(/usr/lib/X11/fvwm/fvwm.bindings)

include(/usr/lib/X11/fvwm/fvwm.styles)

include(/usr/lib/X11/fvwm/fvwm.goodstuff)

include(/usr/lib/X11/fvwm/fvwm.modules)

#####
# Initialization and restart function

Function "InitFunction"
Module "I" GoodStuff
Module "I" FvwmPager 0 1
Exec "I" exec xterm -sb -sl 400 -geometry +75+390 &
Exec "I" xsetroot -solid LightSlateGray
```

```
EndFunction

Function "RestartFunction"
    Module "I"      GoodStuff
    Exec      "I"    xsetroot -solid LightSlateGray
    Module "I"      FvwmPager 0 1
EndFunction
```

system.fvwmrc

fvwm.options

In addition to color and font definitions, the file `fvwm.options` contains a number of other options that determine the look and feel.

```
#####
#
# fvwm.options - general options
#

DeskTopSize 2x2
DeskTopScale 32

# Standard colors
StdForeColor      Black
StdBackColor      #d3d3d3

# Window colors
HiForeColor       Black
HiBackColor       #5f9ea0
StickyForeColor   Black
StickyBackColor   #60c0a0

# Menu colors
MenuForeColor     Black
MenuBackColor     grey
MenuStippleColor  SlateGrey

# Fonts
Font              -adobe-helvetica-medium-r-*-12-*-*-*-*-*
WindowFont        -adobe-helvetica-bold-r-*-12-*-*-*-*-*
IconFont          fixed

# Rectangles in which icons are positioned
IconBox 5 -80 -140 -5
IconBox 5 -160 -140 -85
IconBox 5 -240 -140 -165
IconBox 5 -320 -140 -245

# Motif look and feel
MWMFunctionHints
MWMHintOverride
MWMDecorHints
MWMBorders
MWMButtons

# Moves all windows with contents
OpaqueMove 100

# Suppress automatic desktop change
EdgeScroll 0 0

# Delay in changing desktop section
EdgeResistance 250 50

NoPPosition

# Automatic positioning of new window
```

```
RandomPlacement

# Forces decoration in transient shell
DecorateTransients
```

fvwm.options

The user can create new menus and assign them to a user action.

fvwm.menus

```
#####
#
# fvwm.menus - Menu configuration
#

Popup "Shells"
  Title "Shells"
  Exec "MXterm"      exec mxterm &
  Exec "Color XTerm" exec color_xterm &
  Exec "Rxvt"        exec rxvt &
EndPopup

Popup "Editors"
  Title "Editors"
  Exec "GNU emacs"    exec emacs &
  Exec "NEdit"        exec nedit &
  Exec "Textedit"     exec textedit &
EndPopup

Popup "Graphics"
  Title "Graphics / Viewer"
  Exec "XPaint"       exec xpaint &
  Exec "XV"           exec xv &
EndPopup

Popup "Modules"
  Title "Modules"
  Module "GoodStuff"   GoodStuff
  Module "Identify"    FvwmIdent
  Module "SaveDesktop" FvwmSave
  Module "Pager"       FvwmPager 0 1
  Module "FvwmWinList" FvwmWinList
  Module "FvwmIconBox" FvwmIconBox
EndPopup

Popup "Window Ops"
  Title "Window Ops" "
  Move "&Move Alt+F7"
  Resize "&Size Alt+F8"
  Lower "&Lower Alt+F3"
  Raise "Raise "
  Stick "(Un)Stick "
  Iconify "(Un)Mi&nimize Alt+F9"
  Maximize "(Un)Ma&ximize Alt+F10"
  Maximize "(Un)Maximize Vertical " 0 100
  Nop " "
  Destroy "&Kill Alt+F4"
  Delete "Delete "
EndPopup

Popup "Window Ops2"
  Move "&Move Alt+F7"
  Resize "&Size Alt+F8"
  Iconify "(Un)Mi&nimize Alt+F9"
  Maximize "(Un)Ma&ximize Alt+F10"
  Lower "&Lower Alt+F3"
  Nop " "
  Destroy "&Kill Alt+F4"
```

```

        Delete "Delete "
        Nop    " "
        Module "ScrollBar"      FvwmScroll 2 2
EndPopup

#####
#
# Mainmenu
Popup "Programs"
    Title "Programs"
    Exec  "Xterm"                exec xterm -sb -sl 400
&
    Popup "Shells"              Shells
    Popup "Editors"             Editors
    Popup "Graphics"            Graphics
    Popup "Modules"             Modules
    Exec  "Screen Lock"         exec xlock &
    Nop    " "
    Restart "Restart Fvwm"      fvwm
    Quit   "Exit"
EndPopup

```

fvwm.menus

Within the configuration file of fvwm, new functions can be defined that are usually assigned to a keyboard or mouse action.

```

#####
#
# fvwm.functions - function definition
#
Function "Move-or-Raise"
    Move      "Motion"
    Raise     "Motion"
    Raise     "Click"
    RaiseLower "DoubleClick"
EndFunction

Function "maximize_func"
    Maximize  "Motion" 0 100
    Maximize  "Click"  0 80
    Maximize  "DoubleClick" 100 100
EndFunction

Function "window_ops_func"
    PopUp     "Click"      Window Ops2
    PopUp     "Motion"     Window Ops2
EndFunction

Function "Move-or-Lower"
    Move      "Motion"
    Lower     "Motion"
    Lower     "Click"
    RaiseLower "DoubleClick"
EndFunction

Function "Move-or-Iconify"
    Move      "Motion"
    Iconify   "DoubleClick"
EndFunction

Function "Resize-or-Raise"
    Resize    "Motion"
    Raise     "Motion"

```

```

        Raise      "Click"
        RaiseLower  "DoubleClick"
EndFunction

```

fvwm.functions

The file `fvwm.bindings` contains the mappings between `fvwm.bindings` mouse and keyboard actions and the associated actions.

```

#####
#
# fvwm.bindings - keyboard- and mouse configuration
#
# Structure of a configuration line:
#
#      <key>      <context> <modifier> <function>
#
#      <key>      (mouse) key
#      <context>  R - root window
#                W - application window
#                T - title bar
#                S - window sides
#                F - window frame
#                I - icon
#                A - everything but title bar
#                0,1,2,... - window element
#      <modifier> N - no modifier key
#                A - alternate
#                C - control
#                M - meta
#                S - shift
#                mod1-mod5 - X11 modifiers
#      <function> fvwm function
#
# mouse click on root window
Mouse 1  R      A      PopUp "Programs"
Mouse 2  R      A      PopUp "Window Ops"
Mouse 3  R      A      Module "FvwmWinList" FvwmWinList
Transient
# window element
Mouse 0  1      A      Function "window_ops_func"
Mouse 0  2      A      Function "maximize_func"
Mouse 0  4      A      Iconify
Mouse 1  F      A      Function "Resize-or-Raise"
Mouse 1  TS     A      Function "Move-or-Raise"
# icon actions
Mouse 1  I      A      Function "Move-or-Iconify"
Mouse 2  I      A      Iconify
# window operations
Mouse 2  FST    A      Function "window_ops_func"
Mouse 3  TSIF   A      RaiseLower
# keyboard shortcut
Key F1    A      M      Popup "Window Ops"
Key F2    A      M      Popup "Programs"
Key F3    A      M      Lower
Key F4    A      M      Destroy
Key F5    A      M      CirculateUp
Key F6    A      M      CirculateDown
Key F7    A      M      Move

```

Key F8	A	M	Resize
Key F9	A	M	Iconify
Key F10	A	M	Maximize

fvwm.bindings

fvwm.styles The file `fvwm.styles` establishes the look and feel of individual applications.

```
#####
#
# fvwm.styles - Style configuration
#
Style "*" BorderWidth 7, HandleWidth 5
Style "FvwmPager" Sticky, NoTitle
Style "FvwmBanner" StaysOnTop
Style "GoodStuff" Sticky, WindowListSkip, NoTitle
Style "xterm" Icon terminal.xpm
Style "xcalc" Icon rcalc.xpm
Style "xman" Icon xman.xpm
Style "xvgr" Icon graphs.xpm
Style "Mail" Icon sndmail.xpm
Style "emacs*" Icon editor2.xpm
```

fvwm.styles

fvwm.modules Each of the `fvwm` modules has its own configuration possibilities, which are summarized in the file `fvwm.modules`.

```
#####
#
# fvwm.modules - Module configuration
#
##### Window identifier #####
*FvwmIdentBack MidnightBlue
*FvwmIdentFore Yellow
*FvwmIdentFont -adobe-helvetica-medium-r-*-*-12-*-*-*-*-*

##### FvwmWinList #####
*FvwmWinListBack #d3d3d3
*FvwmWinListFore Black
*FvwmWinListFont -adobe-helvetica-bold-r-*-*-10-*-*-*-*-*
*FvwmWinListAction Click1 Iconify -l,Focus
*FvwmWinListAction Click2 Iconify
*FvwmWinListAction Click3 Module "FvwmIdent" FvwmIdent
*FvwmWinListUseSkipList
*FvwmWinListGeometry +0-1

##### FvwmIconBox #####
*FvwmIconBoxIconBack #cfcfcf
*FvwmIconBoxIconHiFore black
*FvwmIconBoxIconHiBack #5f9ea0
*FvwmIconBoxBack #cfcfcf
*FvwmIconBoxFore blue
*FvwmIconBoxGeometry 1x5+0+89
*FvwmIconBoxMaxIconSize 64x38
*FvwmIconBoxFont
-adobe-helvetica-medium-r-*-*-12-*-*-*-*-*
```

```

*FvwmIconBoxSortIcons
*FvwmIconBoxPadding 4
*FvwmIconBoxLines 10
*FvwmIconBoxPlacement Top Left
#
# mouse bindings
#
*FvwmIconBoxMouse 1 Click RaiseLower
*FvwmIconBoxMouse 1 DoubleClick Iconify
*FvwmIconBoxMouse 2 Click Iconify -1, Focus
*FvwmIconBoxMouse 3 Click Module
"FvwmIdent" ndings
#
# Key bindings
#
*FvwmIconBoxKey r RaiseLower
*FvwmIconBoxKey space Iconify
*FvwmIconBoxKey d Close
#
# FvwmIconBox built-in functions
#
*FvwmIconBoxKey n Next
*FvwmIconBoxKey p Prev
*FvwmIconBoxKey h Left
*FvwmIconBoxKey j Down
*FvwmIconBoxKey k Up
*FvwmIconBoxKey l Right
#
# Icon file specifications
#
*FvwmIconBox "*" unknown1.xpm
*FvwmIconBox "Mosaic" www-shape.xpm
*FvwmIconBox "xterm" terminal.xpm
*FvwmIconBox "GoodStuff" toolbox.xpm
*FvwmIconBox "*ircon*" daffy.xpm
*FvwmIconBox "*anual*" xman.xpm

##### Pager #####
*FvwmPagerBack #908090
*FvwmPagerFore #484048
*FvwmPagerFont -adobe-helvetica-bold-r-*-10-*-*-*-*-*
*FvwmPagerHilight #cab3ca
*FvwmPagerGeometry 0+0
*FvwmPagerLabel 0 Strobel
*FvwmPagerLabel 1 Uhl
*FvwmPagerSmallFont 5x8

```

fvwm.modules

The configuration of the GoodStuff module resides in its own file, named `fvwm.goodstuff`. GoodStuff enables including the most important applications in an icon bar. A click on an icon launches the corresponding program. The Swallow option enables depicting programs like `xload` and `xclock` in the icon bar.

`fvwm.goodstuff`

```

#####
#
# fvwm.goodstuff - Goodstuff . configuration
#
*GoodStuffBack gray60
*GoodStuffGeometry 65x715-1+0
*GoodStuffColumns 1
*GoodStuffFont -adobe-helvetica-medium-r-*-12-*-*-*-*-*

```


#	Name	Icon	Action	WindowTitle	command
*GoodStuff	" "	" "	Swallow	"xclock"	xclock
-bg gray60 &					
*GoodStuff	" "	" "	Swallow	"xload"	xload
-bg gray60 &					
*GoodStuff	" "	" "	Swallow	"xbiff"	xbiff
-bg gray60 &					
*GoodStuff	XTerm	terminal.xpm	Exec	"xterm"	xterm
-sb -sl 400 &					
*GoodStuff	NetScape	www.xpm	Exec	"Netscape"	
netscape &					
*GoodStuff	Xman	xman.xpm	Exec	"Manual Page"	xman
-bothshow -notopbox &					
*GoodStuff	Mail	sndmail.xpm	Exec	"Mail"	xterm
-T Mail -e pine &					
*GoodStuff	Emacs	editor.xpm	Exec	"emacs"	emacs &
*GoodStuff	Exit	lbolt.xpm	Quit		

fvwm.goodstuff

The above classification of the `fvwm` configuration is not absolutely necessary. However, it does add some structure to the `system.fvwmrc` otherwise incomprehensible configuration file `system.fvwmrc`.

Networking

A significant aspect in the discussion of modern workstations and their operating systems centers around networking capability, that is, the possibility to integrate the system into an existing network.

The complete development of Linux would have been impossible without the Internet. Thus even very early versions of the kernel included TCP/IP and the necessary drivers for PC network board s.

This chapter presents the basics of networking and describes the configuration of lower network levels. Since discussing all the details of TCP/IP would lead far beyond the scope of this book, we have simplified some of the issues. For further information we refer to the RFC s (see next page) and the numerous books on TCP/IP and network administration dealing exclusively with this subject.

9.1 Network hardware

Connecting to a network requires little in the way of hardware: two computers can be connected in an elementary local area network (LAN) with two simple Ethernet boards, a piece of thin Ethernet cable, two T-connectors, and two terminal resistors. This makes networking affordable even for private users. Linux supports various pocket adapters and Arcnet boards as well as drivers for nearly all the popular Ethernet boards. You can connect computers at even less cost using a parallel or serial cable and modems. Special protocols like SLIP, CSLIP, PPP and PLIP support such networking.

Internet

basics

RFC

LAN

Ethernet

SLIP; PPP

9.2 TCP/IP

TCP/IP The de facto standard for networking UNIX computers is TCP/IP, a protocol that was developed for the Internet in the early 1970s and has since become available for almost all computer platforms. The history of TCP/IP is closely connected to that of the Internet, so these two subjects are usually treated together.

History

ARPA TCP/IP evolved as a protocol for ARPA NET (later DARPA NET and
Internet then Internet), which at the time primarily interconnected American universities. The network was commissioned and financed by the Advanced Research Project Agency. This U.S. government agency was involved primarily in military projects; hence it is no surprise that TCP/IP was also declared as the standard of the U.S. Department of Defense. Meanwhile the Internet has grown to include many subnetworks worldwide.

RFCs

information TCP/IP is not a standard like those of ANSI, ISO, and IEEE, but a manufacturer-independent definition that is available to everyone in the form of a Request for Comment (RFC). An RFC is usually a description of a protocol or a proposal for a new protocol. Not every RFC is a standard, however. Many have a more informal character. RFCs are available via FTP or e-mail from an RFC
e-mail archive or from many other FTP servers. In Germany, for instance, this is `ftp.uni-stuttgart.de`, where the RFCs are listed in the directory `/pub/doc/standards/rfc`. You can also retrieve RFCs via e-mail from the official RFC archive of the InterNIC (`ds.internic.net`). To do this, you send a message with the contents `help` to the address `mailserv@ds.internic.net`. The server usually responds within half an hour with a description and detailed instructions.

Structure

4 levels TCP/IP essentially consists of four *levels*, which, with some
ISO/OSI layers deviations, correspond to today's standard ISO/OSI Reference Model

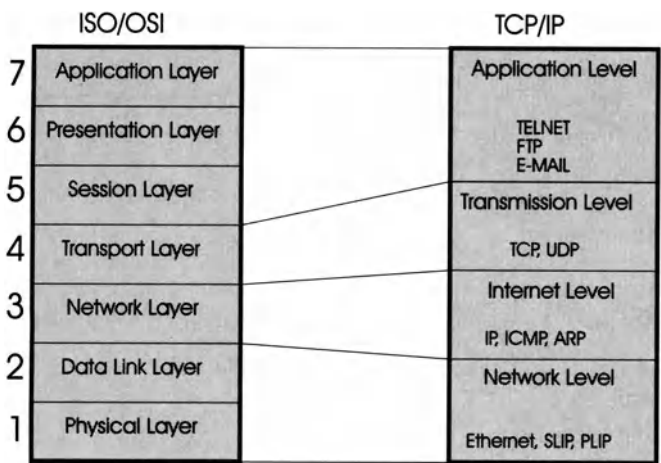


Figure 9.1. Correspondence of ISO/OSI layers and TCP/IP levels

(see Figure 9.1). The lowest level is termed the network level and corresponds to *layers* one and two of the ISO/OSI Reference Model. As a rule, Ethernet is used at this level.

The second level is the Internet level with its Internet Protocol (IP). This level approximates the ISO/OSI layer three. This layer enables the transfer of data packages (datagrams) independently of the actual network that makes the connection. On this level, networking software identifies a computer by its unique IP address. The third level corresponds to ISO/OSI layer four and offers TCP and UDP protocols. TCP stands for Transmission Control Protocol and makes a virtual connection, while UDP means User Datagram Protocol and offers a simple package service.

The fourth level of TCP/IP covers ISO/OSI layers five to seven and is termed the application level. While levels one to three are normally components of the operating system (in Linux they are part of the kernel), level four consists of normal programs. The simplest and most propagated services on this level are Telnet for the virtual terminal, FTP for transferring files, and e-mail for transferring electronic correspondence. Telnet and FTP exist as programs of the same respective names that the user can launch directly. E-mail

is supported from the TCP/IP end with only a simple transfer mechanism that is used by various other programs.

FTP In the following we will describe primarily the lower levels of TCP/IP in greater detail. The next chapter is devoted to Internet and other applications.

9.3 IP

data packets The Internet Protocol (IP) transports data packets between network interfaces, usually of different computers. Data that is sent by a higher protocol, such as TCP, goes to the IP level, where it is packed in IP packets. Based on its routing table, the IP level makes a decision about where to actually send the packets and passes them on to the appropriate network level accordingly. We will illustrate this with an example after the following explanation of the basic concepts.

routing

Addresses

In a network connection with TCP/IP, each network interface of a computer receives an IP address that is unique worldwide. The address consists of four numbers between 0 and 255, separated by periods, for example, 141.7.1.40. This address consists of a network component and a host address. It must be set when the computer is started up.

network and
host address

For small private networks that lack connections to other networks, the IP address is relatively insignificant; users can select them at will or assume the default values suggested by the installation program. Here it is only important to assure unambiguity within the local-area network.

default values

Network classes

The IP addresses are important when setting up an Internet connection. The Network Information Center (NIC) assigns one or more network addresses for the new connection. These addresses belong to one of the classes A, B, or C, (see Figure 9.2) and consist of 1, 2, or 3, bytes, depending on the class. The first bits of an address indicate the class to which the address belongs. The network address constitutes the start of the IP address. The local network administrator

network classes

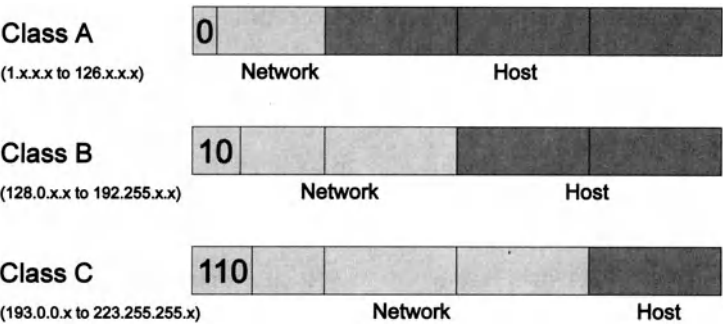


Figure 9.2. Network classes

can assign the remaining bits. In a class C network, the IP addresses consist of an officially determined network address with 3 bytes and a local component with 1 byte. Since addresses ending with 0 or 255 have a special meaning, a class C network can encompass up to 254 IP addresses. Aside from the classes A, B, and C, there are also class D and E addresses. These are reserved for multicasting and extensions.

The local component of the IP address can be either used directly for addressing the individual machine or divided into a subnetwork address and a host address by means of a network mask. However, we do not discuss the use of subnetworks further here. Details can be found in RTC950.

Loopback device

Even for computers that do not actually possess a network interface there is a virtual interface named *loopback*. As its name suggests, everything that is output via this interface is input again directly. This allows TCP/IP connections for a machine to itself. Thus the TCP/IP programs can be used even without a network interface. The IP address of the loopback interface is usually 127.0.0.1.

ARP

We briefly explain addressing at the network level in order to provide better understanding of the following section. An Ethernet example

0 and 255

subnetworks

network mask

virtual interface

loopback address

Ethernet

will illustrate how an IP packet is passed on from one computer to another.

Ethernet addresses	An IP packet must be packed as Ethernet packets and have an Ethernet address on it to be transported on an Ethernet. Ethernet addresses are determined directly in the hardware of the Ethernet boards and are universally unambiguous. While packing the IP packet into an Ethernet packet, the IP level sends an Address Resolution Protocol (ARP) packet as a broadcast to all computers in the same subnetwork. In this way the IP level can determine to which Ethernet address the packet should be sent. You could interpret the ARP broadcast as a question like this: "Please respond if you know the Ethernet address of the interface with the IP address a.b.c.d." If the desired address is in the same subnetwork, the appropriate computer will respond with an ARP reply communicating the Ethernet address. The first computer can then pack the IP packet in a correctly addressed Ethernet packet and send it.
ARP	
ARP broadcast	
ARP reply	

Routing

Not only the Internet, but even large companies and universities do not consist of only a single Ethernet to which all computers are directly connected. Rather, they consist of many heterogeneous partial segments connected via repeaters, bridges, and routers.

repeater, bridge, and router	Since IP should be independent of the network levels used beneath it, packets can only be directly addressed from computer A to computer B, for instance, with an Ethernet address if A and B are in the same physical subnetwork. If two computers are separated by one or more routers, then communication must go via the routers. This means that packets on the network level must be addressed to the router. Every computer needs to know whether the target address is in the same subnetwork when addressing an IP packet. If the target address is in the same subnetwork, the computer can get the Ethernet address with an ARP request and send the packet directly. If the address is in a different subnetwork, then the computer must address the packet to the appropriate router's Ethernet address. In addition, one computer can have several network interfaces and thus be connected to several subnetworks simultaneously. The computer
router	
ARP request	

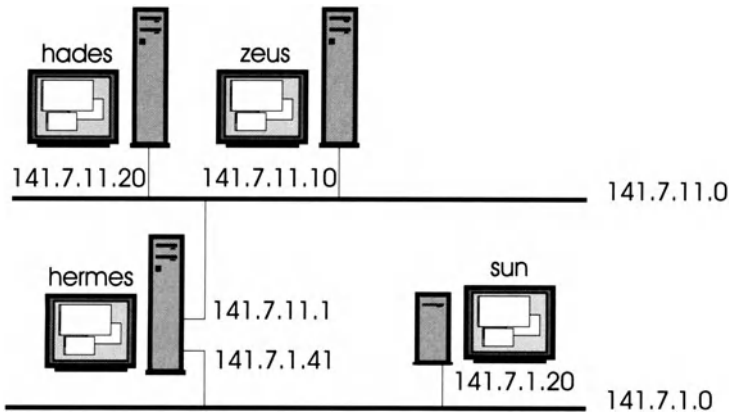


Figure 9.3. Network example

therefore also needs to find out to which interface a particular packet should be sent.

This decision is called *routing*, and it is made by every computer and the designated routers. The IP level is equipped with a table of network or host addresses and an interface, or the address of the next router. The *default route*, a special entry in the table, determines where to send packets without a precise entry.

routing

Example

This example starts with a class B network, which is divided by a network mask into further subnetworks. The officially designated network address is 141.7.0.0, and the network mask used is 255.255.255.0. Three bytes are available for addressing the network and 1 byte for addressing the individual computer in the subnetwork. The configuration of the IP address on startup of the computer determines the network mask. A Linux PC working as a router contains two network boards and is connected to the subnetworks 141.7.1.0 and 141.7.11.0. It establishes the connection between these two subnetworks. The addresses of the respective network interfaces are 141.7.11.1 and 141.7.1.41. Figure 9.3 demonstrates this example.

network mask
network address

The Linux router's routing table contains the following entries:

target network / address	gateway	interface
141.7.11.0	*	eth0
141.7.1.0	*	eth1
default	FH router	eth0

default route

The router is directly connected to the two subnetworks 141.7.11.0 and 141.7.1.0. It cannot reach other IP addresses directly. The default route entry determines that packets for other addresses are passed on to the central FH Heilbronn router. The network mask for its two interfaces, eth0 and eth1, is 255.255.255.0.

routing table

The routing table for the other computer in the 141.7.11.0 subnetwork contains the following entries:

target network / address	gateway	interface
141.7.11.0	*	eth0
default	141.7.11.1	eth0

same subnetwork

This table determines that the computer can send packets only to other computers in the same subnetwork. Other packets are sent to the Linux router at address 141.7.11.1.

If a computer with address 141.7.11.10 sends a packet to the computer with the address 141.7.11.20, then the network masks and the routing entries determine to which address and via which interface the packet will actually be forwarded. The network mask 255.255.255.0 compares only the first three bytes of the IP address with the routing table entries. Thus 141.7.11.20 and 141.7.11.0 match.

target address

The routing entry states that the target address can be reached directly via the interface eth0. If the Ethernet address for 141.7.11.20 is still unknown, then the computer sends out an ARP broadcast to all the other computers in its subnetwork and asks for the Ethernet address for 141.7.11.20. The target computer will respond with its own Ethernet address in an ARP reply. The computer

with the address 141.7.11.10 can then pack its IP packet in an Ethernet packet and send it directly to the receiver.

Ethernet packet

The process becomes more complicated when a packet is sent to a different subnetwork. The sending computer finds an entry in its routing table with a gateway address. Then it must attempt to find out the gateway's Ethernet address in order to send the packet there. The gateway, which is the Linux router in this example, receives the Ethernet packet and unpacks it. When the gateway determines that the IP packet it has received is actually directed to a different address, it attempts to send the packet on using its routing table.

gateway address

Configuration

We describe the configuration of a PC in the subnetwork 141.7.11.0 as in the previous example. First configure the network interfaces with the command `ifconfig`. Then determine the IP address, the network mask, and the broadcast address for each interface. Use the broadcast address to send packets to all the computers in the same subnetwork. In the simplest case and in our example, this is the network address ending with 255 instead of 0, i.e., 141.7.11.255. You will find details on the parameters and the invocation of `ifconfig` on the appropriate manual page.

`ifconfig`

The drivers list the names of the available network interfaces with the booting process. Some common names are, for example, `eth0` for Ethernet, `lo0` for loopback, or `s10` for the first serial interface for SLIP.

interface names

You should use numbers between 2 and 253 for the last byte of addresses for normal computers in a subnetwork. The numbers 1 and 254 are usually reserved for routers, and the numbers 0 and 255 are assigned to network or broadcast addresses.

Once you have configured the interfaces, you must make the entries in the internal routing table of the kernel which records which addresses can be reached via which interfaces. Do this with the command `route`.

routing table

The example below shows the script `/etc/rc.d/rc.inet1`, which makes the necessary settings for a computer with the IP address 141.7.11.10.

`rc.inet1`

```
HOSTNAME=stef1

# Activate loopback
/sbin/ifconfig lo 127.0.0.1
/sbin/route add net 127.0.0.0

# Parameters for this computer:
IPADDR="193.48.121.37"
NETMASK="255.255.255.0"
NETWORK="193.48.121.0"
BROADCAST="193.48.121.255"
GATEWAY="193.48.121.1"

# Activate Ethernet interface
/sbin/ifconfig eth0 ${IPADDR} broadcast ${BROADCAST} netmask
${NETMASK}

# Routing for the local network
/sbin/route add net ${NETWORK} netmask ${NETMASK}

# Routing for other addresses via the router
/sbin/route add default gw ${GATEWAY} metric 1
```

Testing connections

After the network interface has been configured with `ifconfig` and the necessary routing information has been entered, the connection can be test ed. The help program `ping` is frequently used for this. It sends small data packets at regular intervals to the target machine, which then responds.

```
linux0:/home/tul> ping linux1
PING linux1.openconcepts.com (192.0.2.130): 56 data bytes
64 bytes from 192.0.2.130: icmp_seq=0 ttl=119 time=2 ms
64 bytes from 192.0.2.130: icmp_seq=1 ttl=120 time=1 ms
64 bytes from 192.0.2.130: icmp_seq=2 ttl=121 time=1 ms
64 bytes from 192.0.2.130: icmp_seq=3 ttl=122 time=1 ms
64 bytes from 192.0.2.130: icmp_seq=4 ttl=123 time=1 ms
... linux1.openconcepts.com ping statistics ...
5 packets transmitted, 5 packets received, 0% packet loss
round-trip min/avg/max = 1/1/2 ms
linux0:/home/tul>
```

This also allows the assessment of the transmission speed. Network configuration should proceed only after the connection has been tested successfully with `ping`. If `ping` detects problems, the settings of the parameters can be tested and controlled with the commands `ifconfig` and `netstat`. If `ifconfig` is invoked without options, it outputs the status of all the network interfaces. Some Linux distributions use a `netstat` command that recognizes the option `-i`.

```
zeus:/root# ifconfig
lo          IP ADDR 127.0.0.1 BCAST 127.255.255.255 NETMASK
255.0.0.0
           MTU 2000 METRIC 0 POINT-TO-POINT ADDR 0.0.0.0
           FLAGS: 0x004B ( UP BROADCAST LOOPBACK RUNNING )

eth0       IP ADDR 194.45.197.100 BCAST 194.45.197.255 NETMASK
255.255.255.0
           MTU 1500 METRIC 0 POINT-TO-POINT ADDR 0.0.0.0
           FLAGS: 0x0043 ( UP BROADCAST RUNNING )
```

You can display the routing table of the kernel by invoking the command `route` without options or the `netstat` command with the option `-r`.

```
zeus:/root# netstat -r
Kernel routing table
Destination net/address      Gateway address      Flags  RefCnt
Use  Iface
141.7.11.0                    *                    UN      0
47854 eth0
127.0.0.0                     *                    UN      0
0 lo
default                       hermes               UGN      0
4070 eth0
```

9.4 Serial Connections

In order to install a TCP/IP connection via a modem line, use SLIP (Serial Line Internet Protocol) or PPP (Point-to-Point Protocol). This kind of protocol-based system has an advantage over normal modem connections in that it can be used by several users simultaneously.

SLIP, PPP

Figure 9.4 shows our familiar example network with an additional serial modem connection. `zeus` has now taken over the task of the SLIP/ PPP server. It has two interfaces with the same IP address, which is easily possible with a point-to-point connection. Normally, serial interfaces are not configured at system startup, but only when they are actually needed. First you must check whether the kernel in use actually provides PPP or SLIP. The following command outputs a list of the available interfaces:

PPP server

serial interfaces

```
zeus:/home/uhl> cat /proc/net/dev
Inter-| Receive | Transmit
face |packets errs drop fifo frame|packets errs drop fifo colls carrier
lo:   0      0 0 0 0      18443 7 0 0 0 0
ppp0: 1425   1 0 0 0      1406 0 0 0 4 0
ppp1: 0      0 0 0 0      0 0 0 0 0 0
ppp2: 0      0 0 0 0      0 0 0 0 0 0
ppp3: 0      0 0 0 0      0 0 0 0 0 0
sl0:  0      0 0 0 0      0 0 0 0 0 0
```

s11:	0	0	0	0	0	0	0	0	0	0	0
s12:	0	0	0	0	0	0	0	0	0	0	0
s13:	0	0	0	0	0	0	0	0	0	0	0
eth0:	308413	0	0	0	0	287117	0	0	0	28	0
zeus:/home/uhl>											

In the example above, four SLIP devices (s10-s13) and four PPP devices (ppp0-ppp3) are available in addition to one Ethernet device (eth0).

SLIP

SLIP With SLIP, individual IP packets are sent unchanged through the lines. Due to high overhead, this leads to relatively low transfer speeds.

CSLIP For this reason, an improved protocol variant called CSLIP (Compressed Serial Internet Protocol) is used more often now. CSLIP compresses the IP header before transfer, which results in a noticeable improvement in transfer speed.

dip The utility dip is for dialing the telephone number and making the connection to a SLIP router or server. It can be used interactively or controlled from a script. The following example is a simple

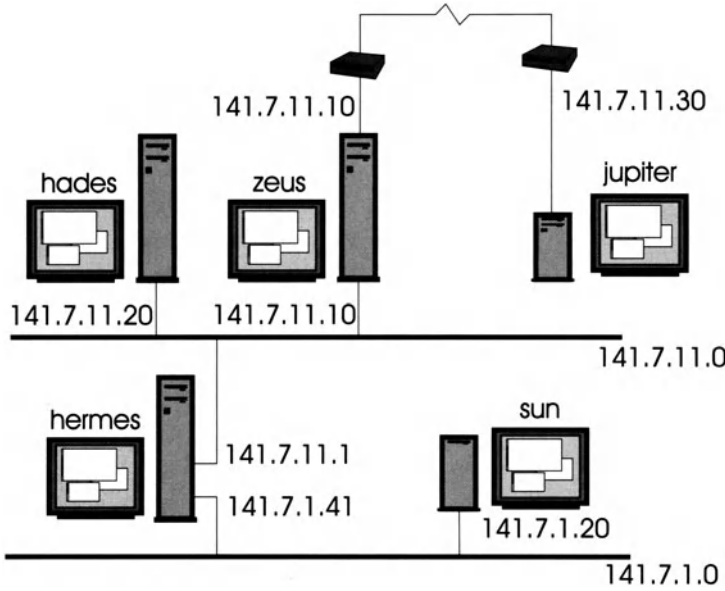


Figure 9.4. TCP/IP compared to a modem

script for dialing a Linux SLIP server. It is invoked with `dip <scriptname>`.

```
# DIP - Login Script (for dip 3.3.7)

main:
# determine IP-addresses
get $local opcon.franken.de
get $remote wuff.mayn.sub.de

# configure port
port cual
speed 38400
reset

# initialize modem
send ATQ0V1E1X4\r

# wait for OK
wait OK 2
if $errlvl != 0 goto modem_trouble

# dial telephone number
dial 993322
if $errlvl != 0 goto modem_trouble

# wait for CONNECT
wait CONNECT 60
if $errlvl != 0 goto modem_trouble

login:
sleep 2

# wait for login message
wait ogin: 20
if $errlvl != 0 goto login_error

# send login name (linus)
send linus\n

# wait for password prompt
wait ord: 20
if $errlvl != 0 goto password_error

# send password (linus)
send linus\n

loggedin:

get $mtu 296
default

done:
print CONNECTED $locip ---> $rmtip
mode CSLIP
goto exit

prompt_error:
print TIME-OUT waiting for SLIPlogin to fire up...
goto error

login_trouble:
print Trouble waiting for the Login: prompt...
goto error

password_error:
print Trouble waiting for the Password: prompt...
```

```
goto error

modem_trouble:
    print Trouble occurred with the modem...
error:
    print CONNECT FAILED to $remote

exit:
```

Once the connection has been established, `dip` automatically configures the SLIP interface and defines a standard routing entry to pass all packets on to the Linux server.

9.5 PPP

PPP The Point-to-Point Protocol (PPP) is another way to establish a TCP/IP connection via modem. Unlike SLIP, PPP has been standardized in several RFCs. In addition, PPP also enables the transfer of other protocols such as Appletalk and Novell's IPX. PPP also has its own authentication protocol, which provides a measure of security with a dial-in connection. Another advantage of PPP is the possibility of dynamic IP address assignment. This means that the server can assign an appropriate address to the client when establishing the connection. Like CSLIP, PPP supports the compression of IP headers, which results in a considerable increase in transfer speed.

Configuration

PPP client The configuration of a PPP client under Linux is similar to that of a SLIP connection and is equally simple. It requires the PPP daemon `pppd` and a program called `chat`. The PPP packets for Linux contain both. The superuser authorization must be running when you install the daemon. The file owner must be `root`, and the SUID bit must be set:

```
jupiter:/root> ls -l /sbin/pppd
-rwsr-xr-x 1 root root 96301 Aug 29 20:02
/usr/sbin/pppd
jupiter:/root>
```

`pppd` completely takes over establishing the connection and directing the transfer of IP packets. Due to the number of options

available, the command line may appear somewhat confusing. We therefore recommend creating a shell script.

command line

```
#!/bin/sh
# pppcall: Establishing a PPP connection

# Telefonnummer
NR=0815

# Login
PROMPT1=ogin:
LOGIN=ppp

# Paßwort
PROMPT2=word:
PASSWD=joshua

MODEM=/dev/ttyS0
SPEED=38400

DIAL=ATDT
PPPD=/usr/sbin/pppd

echo "Connecting to PPP server ($NR) ..."

$PPPD connect "chat -v ABORT BUSY ABORT 'NO CARRIER' '\
    SDIAL$NR CONNECT ' ' $PROMPT1 $LOGIN $PROMPT2
$PASSWD"\
    $MODEM $SPEED -detach crtscts modem defaultroute
```

Another alternative is to enter frequently used options in a file called `/etc/ppp/options`. This file must exist whether or not it is used.

options

```
#
# /etc/ppp/options: Global options of pppd
#
lock # UUCP conform lock file
modem # modem connection
crtscts # hardware flow control
-ddetach # no fork
```

As the script shows, the `chat` command takes care of dialing the remote modem and logging in. The `chat` script can also send character strings to the modem or wait to receive a sequence, so it is similar to the `chat` script for establishing a UUCP connection.

chat script

The `chat` program should also contain the interrupt strings `BUSY` and `NO CARRIER`, as in the script above. This will avoid having to wait long for an interruption when the connection is being structured. In this particular example, `chat` dials the number 0815, then waits for a `CONNECT` response and a log-in prompt. Before a PPP connection is established, `ppp` is given as the log-in name and `joshua` as the

`BUSY, NO CARRIER`

`CONNECT`

password. It may be necessary to make some changes in the script because log-in prompts coming from the server may differ from time to time.

debug option The `chat` and `pppd` debug options are very helpful in looking for configuration errors. Using the `-v` option in `chat` results in a protocol of all activities via the `syslog` daemon. With `pppd`, do this by transferring `-debug`.

dynamic IP address The local IP address is used for a PPP connection on the client end if no other specifications are made. The option `noipdefault` is used to accept a dynamically assigned address from the server.

default route The option `defaultroute` should be transferred as well, so that the default route is automatically set to the PPP server. If you do not wish to assign a default route, the routing can also be explicitly defined.

ip-up To do this, `pppd` invokes the shell script `ip-up` in the directory `/etc/ppp` after the connection has been successfully established. The name of the network interfaces, the path of the serial interfaces, the transfer speed, the local address, and the remote station's IP address are passed on as parameters. Any routes can be set, depending on the transfer parameters, in a script like this.

```
#!/bin/sh
# IP-UP Script for PPP connections
#
case $5 in
    192.7.11.1)      # IP address of the affiliate
        /sbin/route add -net 192.7.11.0 gw 192.7.11.1 ;;
    195.9.12.1)
        /sbin/route add -net 195.9.12.0 gw 195.9.12.1 ;;
esac
```

ip-down The shell script `ip-down` is analogous to `ip-up` and makes it possible to carry out any number of actions after ending a PPP connection.

connection interrupt To interrupt a PPP connection, it suffices to send a termination signal (`TERM`) to the `pppd` with the command `kill` or `killall`. The process ID of the running daemon is in `/var/run/pppN.pid`, whereby `N` is the number of the associated interface. The following command interrupts the connection:

```
zeus:/root# killall pppd
```

If several daemons are running in parallel, then only the correct pppd should be terminated. This can be achieved with the following command:

```
zeus:/root# kill `cat /var/run/ppp0.pid`
```

This ensures that only the pppd responsible for the interface ppp0 is terminated.

Authentication

It makes sense to install additional security measures for a dial-in connection in order to guard against intruders. PPP provides two different authentication protocols for this purpose. The Password Authentication Protocol (PAP) basically acts like a simple log-in mechanism. It checks a user name and an optional coded password at the start. The Challenge Handshake Authentication Protocol (CHAP), which is explained further ahead, provides considerably greater security. CHAP effects an exchange of special information at regular intervals, with which the communicating stations can check each other's identity.

security
authentication
PAP
CHAP

The keyword auth must be added to the file /etc/ppp/options (or to the pppd command line) in order to enforce this kind of mutual authentication. For each connection, distinct secrets, which are matched to each host, can be defined. Depending on the protocol, these are in either /etc/ppp/pap-secrets or /etc/ppp/chap-secrets. A CHAP authentication is always attempted first, then a PAP authentication. If neither of these two files can be found, the connection is terminated.

secrets

A file with CHAP secrets can take the following form:

#			
# CHAP Secrets for zeus.heilbronn.de			
#			
# Client	Server	Secret	Adresse
#			
zeus.demo.de	jupiter.demo.de	"catbox"	zeus.demo.de
jupiter.demo.de	zeus.demo.de	"Eat brown rice, Baby"	jupiter.demo.de
*	zeus.demo.de	"00Schneider"	141.7.11.50

The first column contains the name of the respective remote station. The server column contains the name of the machine where

remote station

the authentication takes place. The third column contains the agreed secret. The final column may optionally contain a host name or an IP address from which the client must log in. This is important for systems allowing any number of clients.

For trouble-free authentication, the client machine must be configured to respond with a fully qualified name (including the domain) when `hostname` is invoked. If `hostname` only responds with a simple name, then the domain can be determined through the `domain` option of the `pppd`.

PPP

Fortunately, the PPP daemon under Linux can also be used to create a PPP server. This requires entering a user in the file `/etc/passwd` that has `ppp`, for example, as a user name.

```
ppp:m3eNH3fgw:200:50:public PPP
account:/tmp:/etc/ppp/ppplogin
```

A script called `ppplogin` is used as a log-in shell and may look like this:

```
#!/bin/sh
#
# ppplogin - Log-in shell for PPP log-in
#
# prevents terminal output via write (1)
mesg n
# turns echo off
stty -echo
# starts pppd in server mode
exec /usr/sbin/pppd -detach silent modem proxyarp crtscts
```

The option `silent` causes the daemon to wait for incoming packets before attempting to establish a connection. `proxyarp` ensures that the server reacts to ARP requests (see section 9.3) concerning the remote station in a point-to-point connection. In this way, a computer connected to a modem can be addressed like a machine in the local Ethernet.

9.6 Parallel connection

The parallel port and a protocol called PLIP (Parallel Line Internet Protocol) provide an economical TCP/IP connection. Two computers are connected with a null printer cable via free parallel ports. This type of network proves particularly attractive for data transfer between a notebook and a workstation.

In principle, the configuration of a PLIP interface works exactly like that of a serial network interface (SLIP, PPP). The name of the first parallel interface is `plip0`. Be sure to choose the PLIP option for the compilation of the kernel (see section 6.2). To configure a point-to-point connection, the option `pointtopoint` and the IP address must be transferred to the remote station when `ifconfig` is invoked.

PLIP

null printer cable

configuration

```
#!/bin/sh
# PLIPUP - init PLIP interface
#
IPADDR="141.7.11.30"      # IP address
NETMASK="255.255.255.0"  # network mask
BROADCAST="141.7.11.255" # broadcast address
POINTOPOINT="141.7.11.10" # point-to-point connection

echo "Setting up PLIP interface..."

/sbin/ifconfig plip0 ${IPADDR} \
                  pointtopoint ${POINTOPOINT} \
                  netmask ${NETMASK} \
                  broadcast ${BROADCAST}

/sbin/route add ${POINTOPOINT} dev plip0

/sbin/route add default hermes.demo.de

echo "PLIP is running."
```

9.7 TCP and UDP

Internet applications rarely use the IP-level routines directly. Instead, they use the TCP or the UDP protocol, which provide more address possibilities and abstraction than IP.

TCP establishes a virtual connection. This means that a connection on the TCP level can be considered a secure data channel. Anything that is sent from one end of the channel can be read on the other end. TCP breaks down the files to be sent and uses IP to send them. Should a transfer error occur or IP packets be lost, the TCP level's task is to repeat the transfer.

virtual connection

packets
checksum

UDP can only send individual packets. It compares the contents of the package with a checksum. If the receiver notices a transfer error, the packet is not sent again, but rather deleted. UDP is not as convenient to use as TCP; however, it is more efficient.

Ports

port number

In addition to the network address, making a connection at this level also requires a port number, which usually corresponds to a particular network service. This permits multiple independent connections via a single network interface. Several of these port numbers are reserved for standard TCP/IP applications: for example, port 23 is for `telnet`, and ports 20 and 21 are for `ftp`. These *well-known ports* are designated in the file `/etc/services`.

Sockets

BSD UNIX
Berkeley sockets

In order to provide a uniform interface for the programming of network applications, the developers of BSD UNIX introduced the Berkeley sockets in the early 1980s. These are a series of kernel routines that are needed to make a connection between two computers and for transferring data between them. Both TCP and UDP connections can be realized via sockets.

Note that it does not matter whether the data transfer takes place within a local-area network (LAN) or in a global wide-area network (WAN). The programming interface is always the same.

The Linux kernel also includes such a socket interface, which makes most well-known UNIX network applications available.

9.8 Host names

symbolic name

Most users do not consider the direct input of an IP address address to be a very comfortable solution. This provided the stimulus for introducing symbolic addresses. Such a symbolic address consists of a host name and a domain name.

host name

The host name in conjunction with the domain name identifies the computer uniquely around the world. Instead of the IP address `141.7.1.40`, for example, a user could enter the symbolic address `linux1.rz.fh-heilbronn.de`. In this case `linux1` is the host

name and `rz.fh-heilbronn.de` the name of the domain, which is a subdomain of `fh-heilbronn.de`.

The correspondence of IP address and symbolic address can be specified locally in the file `/etc/hosts`. Here additional names, or alias es, can be defined for a host. Aliases serve to define shorter names and thus to make the entry of addresses more comfortable. The following is an excerpt from the file `/etc/hosts`:

#IP address	symbolic address	alias
141.7.1.40	linux1.rz.fh-heilbronn.de	linux1
127.0.0.1	localhost	
141.7.1.20	sun1.rz.fh-heilbronn.de	sun1
141.7.1.25	risc11.rz.fh-heilbronn.de	risc1 news newsserver

Excerpt from the file `/etc/hosts`

DNS and name server

Since maintaining hundreds or thousands of entries in this file proves impossible, a hierarchical system of name server s was set up to manage these symbolic names and to automatically exchange addresses among themselves. For each domain, i.e., all machines with a certain domain name, there is a responsible name server. A detailed description of the concepts *domain name*, *subdomain*, and *name server* can be found in RTC1034 and RFC1035.

The UNIX TCP/IP programs convert the host name to the corresponding IP address address by invoking a C library routine. This routine, called `resolver`, reads the file `/etc/hosts` and makes a connection to the next name server if necessary (i.e., if the address is outside its own domain). The order in which this is to occur can be specified in the file `/etc/host.conf`.

```
# /etc/host.conf
order hosts, bind
multi on
```

File `/etc/host.conf`

/etc/hosts

The entry `order hosts, bind` indicates that the file `/etc/hosts` is to be read first. If it contains no entry for the host name in question, then the name server is contacted. The address of the next name server is stored in the file `/etc/resolv.conf`. The line `multi on` means that the resolver should return all valid addresses to a host when multiple addresses in are entered in the file `/etc/hosts`. The On-Line Manual page for `resolv+` describes these and other options in detail.

/etc/resolv.conf

The actual configuration of the resolver is stored in the file `/etc/resolv.conf`. This file also contains the address of the nearest name server, the name of the local domain, as well as possible host name extensions for querying name servers.

```
# /etc/resolv.conf
domain demo.de
nameserver 141.7.11.1
nameserver 141.7.1.25
search demo.de
search beispiel.de
```

search

The line with `search` indicates that for a name server query both `demo.de` and `beispiel.de` should be tried as extensions of the host name. The host name is thus sufficient without the domain name for a computer from either domain, assuming that the host name is unambiguous.

Name server

daemon

NAG

Many Linux installation packages include the name server daemon `named`. The configuration of this name server is no simple matter, however, and for smaller networks a name server usually proves superfluous. For detailed descriptions of how a name server functions and of TCP/IP configuration files, refer to the *Linux Network Administration Guide* (NAG). *DNS and Bind*, published by O'Reilly, contains a very detailed description. This section merely provides a general overview of the files and programs used.

configuration file

The first configuration file of the name server daemon `named` is `/etc/named.boot`. This file determines the (primary) domains for which the server is responsible, the (secondary) domains for which

it should regularly manage the names and addresses, and the files in which the respective tables are stored.

```
; boot file for primary nameserver
; Domain demo.de
;

directory      /etc/bind

; --- definition file for zone of authority
primary        demo.de                        db.demo
;
; reverse mappings for local (.11) subnet..
primary        11.7.141.in-addr.arpa         db.141.7.11
;
; --- file defining localhost
primary        0.0.127.in-addr.arpa          db.127.0.0
;
; --- file to hold cached IN addresses
cache          .                             db.cache
```

File /etc/named.boot

In this example, the individual tables are stored in the directory `/etc/bind`. This directory is also specified in `/etc/named.boot`. The tables can be separated into two types: One type provides addresses and further information on a host name; the other provides the host name of an address. The following examples show sections from each type of table.

```
; Address to hostname mappings for net 127.0.0.1
@      IN SOA hermes.demo.de. dns.demo.de. (
                                1994120600 ; Serial
                                21600      ; Refresh
                                1800       ; Retry
                                3600000    ; Expire
                                86400 )   ; Minimum

      IN      NS      ns.demo.de.
      IN      PTR     localhost.
```

File db.127.0.0

```
; Address to host mappings for 141.7.11
; local subnet demo.de

@      IN SOA hermes.demo.de. dns.demo.de. (
                                1995010302 ; serial
                                43200      ; refresh : 12h
                                1800       ; retry  : 30 min
                                3600000    ; expire  : 41 Tage
                                86400 )   ; minimum : 24h

; nameservers
      IN      NS      ns.demo.de.

; Hosts
```



```

1          IN      PTR      hermes.demo.de.
10         IN      PTR      zeus.demo.de.
20         IN      PTR      hades.demo.de.
30         IN      PTR      jupiter.demo.de.

```

File db.141.7.11

```

;
;  demo.de
;  host to address mappings
;
@           IN SOA      hermes.demo.de. dns.demo.de. (
                        1994010402 ; serial
                        43200      ; refresh : 12h
                        1800       ; retry  : 30 min
                        3600000     ; expire  : 41 Tage
                        86400 ) ; minimum : 24h

; nameservers
;
;  IN      NS      hermes.demo.de.
; mx record for demo.de
;  IN      MX      10 hermes.demo.de.
;
;  Hosts
hermes      IN      A      141.7.11.1
            IN      HINFO  "i486" "Linux"
            IN      MX      10 hermes.demo.de.
news       IN      CNAME   hermes.demo.de.
ftp        IN      CNAME   hermes.demo.de.
www        IN      CNAME   hermes.demo.de.
ns         IN      CNAME   hermes.demo.de.
;
zeus       IN      A      141.7.11.10
            IN      HINFO  "i486" "Linux"
            IN      MX      10 hermes.demo.de.
;
hades      IN      A      141.7.11.20
            IN      HINFO  "i486" "Linux"
            IN      MX      10 hermes.demo.de.
;
jupiter    IN      A      141.7.11.30
            IN      HINFO  "i486" "Linux"
            IN      MX      10 hermes.demo.de.
;

```

File db.demo

db.cache File db.cache has particular significance. It stores the addresses of the root name server. Its contents rarely change.

```

; Root Nameserver Cache
.          99999999 IN NS  NS.INTERNIC.NET.
.          99999999 IN NS  AOS.ARL.ARMY.MIL.
.          99999999 IN NS  NS1.ISI.EDU.
.          99999999 IN NS  C.PSI.NET.
.          99999999 IN NS  TERP.UMD.EDU.
.          99999999 IN NS  NS.NASA.GOV.
.          99999999 IN NS  NIC.NORDU.NET.
.          99999999 IN NS  NS.ISC.ORG.
NS.INTERNIC.NET. 99999999 IN A 198.41.0.4
AOS.ARL.ARMY.MIL. 99999999 IN A 128.63.4.82
AOS.ARL.ARMY.MIL. 99999999 IN A 192.5.25.82

```

NS1.ISI.EDU.	99999999	IN	A	128.9.0.107
C.PSI.NET.	99999999	IN	A	192.33.4.12
TERP.UMD.EDU.	99999999	IN	A	128.8.10.90
NS.NASA.GOV.	99999999	IN	A	192.52.195.10
NS.NASA.GOV.	99999999	IN	A	128.102.16.10
NIC.NORDU.NET.	99999999	IN	A	192.36.148.17
NS.ISC.ORG.	99999999	IN	A	192.5.5.241

File db.cache

Finding errors

The program `nslookup` is frequently used to look for errors involving name servers. With this program, a name server can be queried interactively. The program can be invoked either with a name in the command line or without arguments. If no arguments are given, `nslookup` responds with a prompt, and the user can enter special commands and search names. The command `help` displays the most important commands.

9.9 UUCP

UUCP stands for UNIX to UNIX Copy and probably represents the oldest possibility of connecting UNIX machines to one another via a network. Establishing the connection normally takes place between the individual machines using a serial cable and a modem. A UUCP network can transfer any amount of data and commands. In general, however, it provides the most economical mail and news link.

UUCP works on the principle of *data forwarding*. This means that in one session it transfers first a data packet and then a command to process the transferred data on the remote machine. When mail is transferred, for instance, the command `rmail` is transferred along with the actual contents of the mail. Data are not usually transferred immediately as they occur, but rather only at certain times (batch system). This keeps telephone costs as low as possible by taking advantage of low rate times. Data are copied into a special directory as they occur and then deleted after successful transfer.

UUCP not only allows the transfer of data between two directly connected computers, but, based on the principle of *store and forward*, the transfer can also be carried out over several stations. Here, however, the exact transfer route must be known. This

possibility is no longer used as widely as it was in the early phases of UUCP use. Since the administration of routes with UUCP maps is quite complicated for larger networks, people usually prefer IP-based transfer now. UUCP is currently used primarily to transfer data from a server connected to the Internet through TCP/IP to a smaller system without an on-line connection.

grade UUCP assigns every job a *grade* according to its importance. Grades are defined with a character in the range 0-9, A-Z and a-z, with 0 having the highest priority. Mail usually has a grade of B or C, news a grade N. The commands `uux` and `uucp` can define the desired priority with the option `-g`.

protocols Data transfer can be carried out through various protocols. Not every UUCP package understands all variants. The `g` protocol is the oldest and most widespread. System V favors the `g` variants. Taylor UUCP also recognizes an extremely fast, bidirectional `i` protocol.

Taylor UUCP This, however, can only be used between Taylor systems. The Taylor UUCP packet consists of a series of commands. The following briefly explains the tasks of these commands.

- connection** • **`uucico`** —establishes a connection to another UUCP system, sends and receives the occurring data, and subsequently invokes the processing of the accompanying commands, such as `rmail` or `rnews` via `uuxqt`.
- commands** • **`uuxqt`** —carries out the commands required via `uux` on a remote system.
- copy** • **`uucp`** —enables copy ing data between any UUCP systems.
- commands** • **`uux`** —enables execution of certain commands on a remote system. In addition to the command name, data that are to be used in the target system may also be transferred.
- jobs** • **`uustat`** —lists current UUCP jobs. Jobs that are not yet completed may also be removed from the UUCP queue.
- **`uuname`** —outputs all known local UUCP systems.
- log files** • **`uulog`** —outputs the contents of the UUCP log file. The output can be limited to certain systems or users.
- interactive** • **`cu`** —establishes an interactive connection to a remote UUCP system.

Configuration

A series of various UUCP packages has developed over the years; they differ primarily in their configuration. The most comfortable UUCP variant is the Taylor UUCP, which is available for free. It not only supports the configuration modes of the antiquated V2 and the HDB, but also its own particular variant, the Taylor mode, which should be used. Old configuration files can be transformed into this format with a converter. In the following we therefore discuss only configuration in the Taylor mode. We assume that all the configuration files may be found in the directory `/usr/lib/uucp`. This path can be defined upon compilation. The following files are found under this path:

`config, sys, port, dialer`

Taylor UUCP
V2, HDB

The name of the local system is defined in the file `config`:

configuration files
`config`

```
# Name of the local UUCP system
#
host name      kirk
```

A neighboring UUCP system must be made known to the local computer in order to establish a connection between the two. This is achieved with the `sys` file, which can contain a series of options in addition to the name of the system, its telephone number, log-in name and password. Such a file looks like the following:

`sys`

```
#
# Global settings for all systems
# -----
#
#
# Systems may be contacted any time
#
time          any
#
# After a successful connection wait 15 minutes (900 seconds)
# before establishing a new connection
#
success-wait   900
#
# ----- gallien -----
# 'gallien' is the name of the UUCP feed.
#
system gallien
#
#
call-login asterix
```

```
call-password obelix
#
# Telephone number
#
phone 0713566354
#
# 'gallien' supports the bidirectional i-Protocol
#
protocol-parameter i packet size 1024
#
# serial port
#
port zyxel
```

port The parameters, such as the name and transfer speed of the port, are defined in a separate port file.

```
#
# ZyXEL modem
# -----
#
# Name of the port
port zyxel
# Modem is connected to port 'ttyS1'
device /dev/ttyS1
# Transfer speed between modem and computer
speed 38400
# Dialer
dialer zyx-fast
```

If the connection is established via a modem, this requires a dialer dialer file containing all the data relative to the modem.

```
#
# Zyxel Modem
#
# Name of the dialer from the port file
dialer zyx-fast
#
# chat string
#
# \T -> Send telephone number
# \r -> Send carriage return (CR)
# \c -> Suppress carriage return at end of string
# \d -> Delay of 1 Second
# \s -> Send space
#
chat "" AT&K4&N17 OK ATDP\r\c CONNECT
#
# Error strings. Chat is terminated, as soon as one of the
# strings is recognized. #
chat-fail BUSY
chat-fail NO\SDIALTONE
chat-fail NO\SCARRIER
#
# after successful connection
#
complete \d\d++\d\dATH0Z\r\c
#
```

```
# after connection interrupt
#
abort          \d\d++\d\dATH0Z\r\c
```

Log files

All the activities of the UUCP system are recorded in several log files. These are also very helpful for configuration. The file `/var/spool/uucp/Log` contains general protocol information; the file `/var/spool/uucp/Stats` contains statistics on the transfer speeds that have been achieved (also see `uulog`). If `uucico` is started with the command line option `-x` in debug mode, then a detailed debug protocol is created under `/var/spool/uucp/Debug`.

`uulog`

Automatic connections

The UUCP user should create a separate `crontab` (see section 2.9) to automate establishing the connection. The `cron` daemon can also facilitate repeat dialing if a line is busy. It is useful in this case to have `uucico` not allow multiple execution within a set interval following a successfully established connection. For instance, to establish a connection to the system `gallien` daily between 7:00 p.m. and 7:15 p.m., the `crontab` below suffices.

`crontab`

```
# (root.crontab installed on Wed Sep 7 18:23:50 1994)
# (Cron version -- $Header: crontab.c,v 2.2 90/07/18 00:23:56
# vixie Exp $) SHELL=/bin/sh
#
# mail any output to 'uucp' no matter whose crontab this is
MAILTO=uucp
#
# daily call to system 'gallien' after 7:00 p.m.
0,3,6,9,12 19 * * *           /usr/lib/uucp/uucico -s
gallien
#
# abbreviate the UUCP Log files by 20.00
#
00 20 * * *           /var/lib/smmail/savelog -m 300 -c 2 -t
/var/spool/uucp/Log
00 20 * * *           /var/lib/smmail/savelog -m 300 -c 2 -t
/var/spool/uucp/Stats
```

The length of the restart interval in this case would have to be set at 15 minutes (`success-wait 900`). The other two `crontab` entries serve to shorten the constantly growing UUCP log files.

`restart interval`

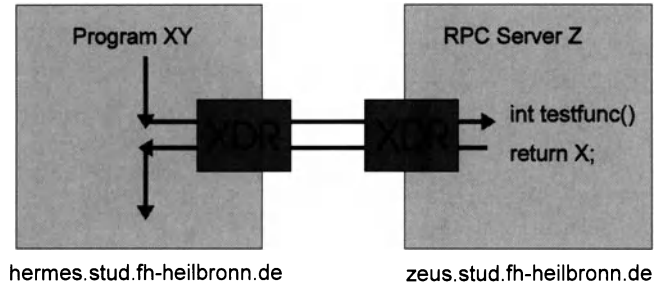


Figure 9.5. RPC invocation

9.10 RPC

Some network services, including the Network File System NFS, are based on the Remote Procedure Calls (RPC) from Sun. This refers to a mechanism allowing execution of individual routines on a remote computer in the network. There is an RPC server, which provides the subprograms, and clients, which use parameters to invoke these subprograms (see Figure 9.5). RPC is thus a special type of communication between processes via a network. Unlike network programming through sockets, in this case one communication channel is abstracted.

The eXternal Data Representation (XDR), a machine-independent type of data representation, is usually used together with RPC to exchange data between computers with differing processor architectures. One frequently occurring difference between various processors, for example, is the representation of integral numbers. The XDR description allows the definition of an RPC routine's parameters, so that they can be converted through the appropriate subroutine between the machine-independent format and the internal format of an individual computer.

The program `rpcgen` allows automatic invocation of these routines and the internal RPC routines. It generates C source code for the RPC server and the client from a formal description of the RPC routines.

In order to establish a connection to an RPC server on a different machine, the `portmap` daemon must be running

on that machine. This daemon recognizes the available services, and it forwards the RPC invocation from the network to the appropriate RPC server. The information on registered programs that the `portmap` daemon has stored can be queried for diagnostic purposes with the `rpcinfo` program.

9.11 NIS

Since the consistent management of accounts and their passwords on a network can be quite arduous, Sun Microsystems developed the Network Information System (NIS). Instead of storing user information, available network services, and other system configuration information on each machine, a central NIS server manages these data. If the administrator creates a new user or if a user changes a password, NIS handles the necessary forwarding of this information to all affected machines.

configuration

The NIS client software is already included in many distributions or can be drawn from most FTP servers, for example, from the directory `/pub/linux/local/yp` on the server `ftp.uni-paderborn.de` (University of Paderborn, Germany).

Paderborn

9.12 NFS

The Network File System (NFS) makes it possible to mount file systems that are released (exported) by other computers, as a part of the local machine's own file systems. This provides transparent access to the directories of the remote computers.

Network File System

The example below shows access to files in the `/home` directory of a remote computer (`stef1`) on the network. Initially, the directory `/stef1` on the machine `dirk1` was empty. After the mount procedure, this directory contains (quasi) all files of the directory `/home` of computer `stef1`. The following is an example of an NFS mount procedure.

mounting

```
dirk1:~# ls
bin/          install/      lost+found/   stef1/        var@
dev/          lastlog@     mnt/          tmp/          vmlinux
etc/          lib/         proc/         user/
vmlinux.old
```



```
home/          linux@          root/          usr/
dirkl:/# ls stefl
dirkl:/# mount stefl:/home /stefl
dirkl:/# ls stefl
dirk/          fritz/          ftp/          peter/       root/       stefan/
dirkl:/#
```

Example of an NFS mount.

NFS was developed by Sun Microsystems. Because Sun released the definitions of the protocol, many other manufacturers were able to integrate NFS into their operating systems. Thus NFS became a standard that asserted itself on almost all platforms even though it was not controlled by any higher authority. NFS is available for almost all UNIX variants, MS-DOS, and other operating systems.

stateless server

A significant feature of NFS is the *stateless server*. This means that the NFS server that exports directories does not store any state information on the clients, but only carries out simple read and write operations. For example, if an NFS server needs to be restarted for whatever reason while an NFS client is copying a file from a directory exported by the server, the client's copying procedure is not interrupted; instead, the client waits until the server responds again and then continues the copying procedure. This procedure becomes problematic when it comes to the synchronization of access by multiple clients who want to write to the same file.

parallel access

Another drawback of NFS is the security of its protocol in terms of privacy and unauthorized access. Newer distributed file systems such as the Andrew File System (AFS) or the Distributed File System (DFS) prove superior to NFS. DFS is part of the Distributed Computing Environment (DCE) of the Open Software Foundation (OSF). However, neither AFS nor DFS has attained the level of proliferation that NFS enjoys.

AFS

Linux also provides a network file system. However, the respective drivers must have been compiled into the kernel. Then at run time the `portmap` daemon as well as the `rpc.nfsd` and `rpc.mountd` daemon s must be running to allow the use of NFS. The NFS daemon `rpc.nfsd` replies to read and write requests from NFS client computers. The daemon `rpc.mountd` is responsible for the mounting information itself. It manages directories and checks the client's authorization for a mount requests.

kernel

nfsd and mountd

This daemon's most important configuration file is `/etc/exports`, which lists all directories that can be mounted by other machines via NFS, along with their respective permissions. Modifications in this file become effective only after both `rpc.nfsd` and `rpc.mountd` are restarted. Here is an example of this file:

configuration file

```
#
# Exported directories
#
/                linux1(rw)
/home            141.7.1.49(rw)
/home/prog       riscl(rw)
```

9.13 LAN manager

One of the most frequently used network protocols in the PC world is the SMB protocol of the LAN manager. Both IBM and Microsoft offer servers under OS/2 and Windows NT, respectively. The biggest drawback of these solutions is the high price of this software. To overcome this problem, a project was started on the Internet to develop a UNIX-based LAN manager server called Samba. This freely available server can be used both as a file server and as a print server.

SMB

Samba

Since Samba runs as a normal process in user space, it does not require any manipulations of the kernel. Since a UNIX kernel normally only recognizes TCP/IP, Samba is also not able to support the actual SMB protocol. Instead it packs individual SMB packets into an IP packet. In principle, any number of network protocols can be sent via an IP network infrastructure this way. By using this trick, this LAN manager protocol, which is not capable of routing, can also be used for a WAN.

SMB in IP

A TCP/IP stack must be installed on the client end, however. Fortunately, Microsoft provides the appropriate software for DOS and Windows free of charge on an FTP server (ftp.microsoft.com). The TCP/IP stack is integrated into Windows 3.11 via the network setup (see Figure 9.6).

TCP/IP stack

Server installation

Once the Samba packet is ready as source code, the two daemons `smbd` (LAN manager server) and `nmbd` (name server) need to be

compilation

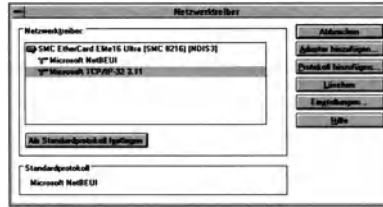


Figure 9.6. TCP/IP stack under WfW 3.11

compiled. First activate the respective locations for Linux in the makefile. Copy the programs created into the directory `/usr/sbin`, where most of the other network daemons are. Since both daemons can be started via the Internet daemon, make an entry in the file `inetd` `/etc/inetd.conf` accordingly.

```
#
# Samba Lanmanager-Server
#
netbios-ssn stream tcp nowait root /usr/sbin/smbd smbd
netbios-ns dgram udp wait root /usr/sbin/nmbd nmbd -G
WORKGROUP
```

entry in `/etc/inetd.conf`

configuration

Server configuration is carried out by a corresponding ASCII file, as usual under UNIX. This file's search path is defined on compilation. The example configuration that accompanies the Samba packet should be used as the basis for custom settings. A public account provided there enables access to the UNIX-end home directory of the individual user as well as the use of UNIX printer services. The following excerpt from the configuration file is responsible for this form of server access:

```
;
; Example configuration of the SMB server
;
[global]
    print command = /usr/bin/lpr -r -P%p %s
    lpq command = /usr/bin/lpq -P
    printer name = lp
    printcap name = /etc/printcap
    guest account = nobody
    password level = 1

[homes]
```

```

comment = Home Directories
read only = no
create mode = 0750
print ok = yes

[printers]
comment = All Printers
path = /usr/spool/public
printable = yes
public = yes
writable = no
create mode = 0700

```

Simple SMB server configuration

A Samba configuration file is divided into several sections that each define a server service. The sections `global`, `homes`, and `printers` have special functions. The individual sections consist of a series of attributes with assigned values that define the characteristics of a service. A series of global parameters or default values is defined in the `global` section.

If a home section exists, then a service is created, which allows all users known to the server to log in from a client and access the home directories. This means that it is not necessary to declare an individual log-in service for each user. The server takes the password needed for logging in from the file `/etc/passwd`. The `path` variable is set to the corresponding home directory of the user.

The printer section is similar to the `homes` section, except that it refers to printer services rather than log-in services. This way, clients can access all printers (`/etc/printcap`) known to the system. A particularly interesting feature is that a PostScript printer queue is provided on the server machine. A corresponding `lpr` filter and Ghostscript make this possible.

In this way, the Linux system can be turned into an economic Raster Image Processor (RIP), even if no PostScript printer is available. Before a file can be printed, it is copied to the server. You can find it in the `/var/spool/public` directory.

The list below explains all the attributes that can be modified for a service. The list differentiates between global attributes (G) and service attributes (S). The appropriate default settings are given in parentheses.

allow hosts	GS	List of clients allowed to use a service; wildcards are permissible: *.fh-heilbronn.de, 192.0.2.*
available	S	Client may use service (yes)
copy	S	Copy of the definition of the previous service
create mask	GS	Mask when a new file is created
create mask	G	Standard mask when a new file is created
dead time	G	Time in minutes until an inactive connection is interrupted
debug level	GS	Debug level
default service	G	Standard service for requesting an unknown service
deny hosts	GS	List of machines prohibited from accessing server
dfree command	G	Script for determining free memory
dont descend	S	List of directories that should appear empty on the client end (none)
getwd cache	G	Cache of current directory (yes)
guest account	G	Standardized user name for guest services
guest ok	S	Guest access available to any users (no)
guest only	S	Exclusively for guest access (no)
keep alive	G	Send a keep alive packet every n minutes
lock directory	G	Path for lock files
locking	G	Control locking (yes)
lpq command	GS	Access path on lpq
mangled names	G	Replace UNIX file names (yes)
map hidden	S	Execute bit should be set for hidden files (no)
map system	S	Execute bit should be set for system files (no)

max connections	G	Maximum number of simultaneously active connections
max xmit	G	Maximum size for transferred packets
only user	GS	Controls whether only registered users have access
password level	G	Maximum number of capital characters in password.
path	GS	Path of respective services
print command	GS	Command for output of transferred printer files
print ok	S	Printer access allowed (no)
printcap name	G	Path of the <code>printcap</code> file
printer name	S	Name of the default printer
protocol	G	Protocol version used
read only	S	Service allows only readable access (yes)
read prediction	G	Enables prediction reading
read raw	G	Reads data in large packets
root directory	G	Server performs <code>chroot</code> on transferred directory
set directory	S	User may use <code>setdir</code> command to change directories (no)
username	S	Name of server user
wide links	G	Allows following any links
write raw	G	Writes in large packets

SMB service attributes

With the Samba server, a PC client can also easily use other server resources, such as a CD-ROM drive or a central removable hard disk drive. CD-ROM

Client configuration

A PC client has access to a Samba LAN manager server via access paths in the following form:



Figure 9.7. WfW connection

```
\\<Server Name>\<service>
```

To access the directory `/home/pcuser` on the server, the access path must look like the following:

```
\\master\pcuser
```

You can reach the printer `ps` of the server `phoenix` like this:

```
\\phoenix\ps
```

Under Windows for Workgroups, the configuration of this kind of server connection is carried out in the file or printer manager, as Figure 9.7 shows.

An appropriate password must be entered the first time the server is accessed.

9.14 PC/NFS

Not only does NFS work well for connecting UNIX computers, it can also be used to integrate DOS PCs into a UNIX network. Aside from the numerous commercial products, you can also use a shareware package (`xfs/xfs32`). As with Samba, the TCP/IP stack from Microsoft is a prerequisite for `xfs32`. The PC/NFS daemon (`pcnfsd`) must be running on the server end in addition to the

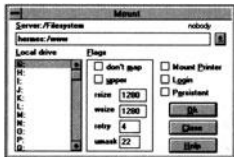


Figure 9.8. Mounting a directory via NFS

NFS and mount daemons. The former is primarily needed for user authentication and printer administration. Most Linux distributions already contain these. They are usually activated at startup in one of the `rc` startup scripts (see section 7.2). A spool directory for printer jobs is transferred as a parameter. Outstanding printer jobs are stored here.

```
/usr/sbin/rpc.pcnfsd /var/spool/xfs
```

With PC/NFS, you can access files and reach printer queues known to the server. A PC/NFS server is configured for access permission via the file `/etc/exports_` (see section 9.12), as usual for NFS. After installation on a PC client, an exported directory can be mounted in the file manager under the new menu selection `Xfs32`.

printer

Access to the Linux printer queue is carried out according to the same principle (see Figure 9.9).

menu

printer queue

The authentication can take place during the mounting or later through a special dialog in the file manager (see Figure 9.10).

authentication

Without authentication, the user has access only to the public directories, because the user ID `-2` is assigned to the user from the

public directories

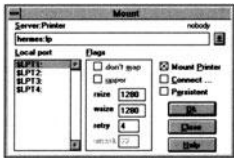


Figure 9.9. Access to the Linux printer queue

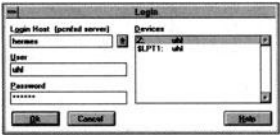


Figure 9.10. Authentication

server side. The user assignment can be subsequently modified at any time.

9.15 Columbia Appletalk (CAP)

Macintosh The Linux server services are not restricted to the integration of ordinary PCs. Apple Macintosh computers can also be served with the help of the Columbia Appletalk Packet (CAP). CAP provides several variants for integration. The most interesting of these, however, is probably the direct Ethershare support.

printer Aside from file and printer serving, CAP also permits access to the Apple Ethershare printer. Installation requires a Linux kernel with a version number of 1.1.70 or higher.

9.16 ISODE

ISO/OSI As explained in Section 3.2, a dedicated ISO standard covers the structure and implementation of networks. TCP/IP was already in existence when this standard was defined. Although TCP/IP can be mapped to the ISO/OSI model, it is not compatible with the standard. A developer who wants to create OSI-compatible applications under Linux can use the freeware ISO/OSI stack named ISODE (by Marshall T. Rose), which has been ported to Linux. This freeware can be found, for example, on the server `sunsite.unc.edu` in `/pub/Linux/system/Network/isode`.

9.17 Novell

IPX To provide connectivity to Novell networks (IPX protocol), there are several approaches to providing transparent access to netware drives.

Under Linux there is as yet no true Netware file system for mounting a Novell server directly under Linux. Novell servers can be accessed, however, in the DOS emulator (see section 4.1). Another possibility is using an NFS add-on packet for the Novell server, with which the server exports its directories via NFS.

DOS emulator

Network Applications

In the previous chapter we explained the theoretical basis of TCP/IP and the lower protocol levels. We will now discuss the programs and servers that use these lower levels.

10.1 Network daemons

As we have already mentioned, the operating system core does not contain most TCP/IP server services. These are realized with separate daemons. Among these are servers for Telnet, FTP, and e-mail, in particular. The list below provides an overview of the most important network daemons available under Linux. Depending on the distribution, the names sometimes have the prefix `rpc.` or `in.` These prefixes were omitted in the list.

server
daemons
Telnet, FTP, mail

- **bootpd** —needed for booting diskless workstations and X terminals.
- **fingerd** —enables the user to finger other users, who are active on a (different) system.
- **ftpd** —used for transferring data from one system to another with FTP.
- **gated** —implements routing protocol for dynamic routing.
- **httpd** —WWW server daemon.
- **identd** —User Identification Server implements the protocol defined in RFC1413 for identifying a user to a connection.
- **imapd** —imap server. This is used for access to mailboxes with imap clients such as pine.

- `ipop2d` and `ipop3d` —Server for POP2 or POP3 Protocol for accessing mails.
- `lpd` —Printer daemon includes the possibility to access a printer from a remote computer.
- `mountd` —permits using a file system of another computer in the local system.
- `nfsd` —makes data available as NFS server.
- `nmbd` —Netbios name server; has nothing to do with the DNS name server.
- `nntpd` —delivers News from Usenet.
- `ntalkd` —server for a `talk` variant.
- `pcnfsd` —server for PC NFS allowing PCs to access a system's files and printer.
- `pppd` —daemon for the PPP protocol (see section 9.5).
- `rlogind` —enables login from a remote system using an `rlogin` command.
- `routed` —also responsible for dynamic routing; can be used instead of `gated`
- `rplayd` —server for the `rplay` command for playing sounds.
- `rshd` —permits executing a command from a remote system.
- `rstatd` —server for `rstat`; outputs the kernel's statistical data.
- `rusersd` —server for the `rusers` command; provides information on the users that are logged in.
- `rwalld` —server for the `rwall` command; outputs announcements to users.
- `rwhod` —server for `rwho`; provides and collects data on users that are logged in.
- `sendmail` —sends and receives mail in the network; `smail` can be used as an alternative.
- `smail` —sends and receives mail in the network. However, it is also important without a direct IP connection; for instance, in a UUCP configuration.
- `smbd` —the SMB/LAN manager server (see section 9.13).
- `talkd` —enables interactive communication with other users through the command `talk`.
- `tcpd` —TCP wrapper daemon. Actual servers can invoke it and check the address and permissions of the clients.

- telnetd —similar to rlogind; enables a user to log in from a remote computer.
- tftpd —used like bootpd for booting other machines in the network.
- timed —time synchronizer daemon; synchronizes the local computer’s time with that of the other computers in the network.
- xntpd —another time daemon; implements the NTP protocol as defined in RFC135.

10.2 Internet daemon (inetd)

Most TCP/IP daemons are activated only when there is actual demand for such services from another computer. If they were always active in the background like other daemons, they would unnecessarily consume memory and computation time. Therefore, these daemons are not automatically started with the booting of the system.

A UNIX system normally provides an Internet daemon (Internet superserver), which waits for messages from the network and is always active, unlike other network daemons. Every service has a fixed port number. The file /etc/services contains a registry of services and their respective port numbers. Another file (/etc/inetd.conf) lists the respective daemons that offer the requested service. Only when an actual connection is requested does inetd start the respective daemon that takes over the connection. Upon termination of the connection, the daemon is also ended, while inetd continues to run.

daemons

Internet daemon

port number

inetd.conf

tcpmux	1/tcp	# TCP Port Service Multiplexer
rje	5/tcp	# remote job entry
echo	7/tcp	
echo	7/udp	
discard	9/tcp	sink null
discard	9/udp	sink null
systat	11/udp	users
systat	11/tcp	users
daytime	13/udp	
daytime	13/tcp	
daytime	13/udp	
netstat	15/udp	
netstat	15/tcp	
qotd	17/udp	quote
quote	17/tcp	# quote of the day
chargen	19/tcp	ttytst source
chargen	19/udp	ttytst source
ftp-data	20/tcp	
ftp	21/tcp	

telnet	23/tcp	
smtp	25/tcp	mail #Simple Mail Transfer
nsw-fe	27/tcp	# NSW User System FE [24, RHT]

Section of a file /etc/services

These files are normally not changed. They only need to be modified if new services are added or if new options become necessary with a daemon update.

telnet	stream	tcp	nowait	root	/etc/telnetd	telnetd
ntalk	dgram	udp	wait	root	/etc/ntalkd	ntalkd
ftp	stream	tcp	nowait	root	/etc/ftpd	ftpd -l
finger	stream	tcp	nowait	root	/etc/fingerd	finger
shell	stream	tcp	nowait	root	/etc/rshd	rshd
login	stream	tcp	nowait	root	/etc/rlogind	rlogind
tftp	dgram	udp	wait	root	/etc/tftpd	tftpd /home/ftp
# Internal to inetd				.		
echo	stream	tcp	nowait	root	internal	
echo	dgram	udp	wait	root	internal	
discard	stream	tcp	nowait	root	internal	
discard	dgram	udp	wait	root	internal	
daytime	stream	tcp	nowait	root	internal	
daytime	dgram	udp	wait	root	internal	
chargen	stream	tcp	nowait	root	internal	
chargen	dgram	udp	wait	root	internal	

The file /etc/inetd.conf

10.3 Telnet

The Telnet protocol is one of the oldest in the Internet. It enables logging in a user to log in to other computers in the network. This means that all the client's keyboard entries are sent to the server and all screen output is sent from the server to the client. The client thus simulates a virtual terminal.

Under Linux and most other UNIX variants, Telnet is implemented by the server daemon telnetd and the client program telnet. Telnet client programs also exist for most other operating systems, including even DOS.

To log in on a different computer, the user invokes the program with a computer name or an IP address. A TCP connection is then established with the Telnet daemon of the computer named.

```
linux2:/home/stefan>telnet sun1
Trying 141.7.1.20...
Connected to sun1.
Escape character is '^]'.
```

```
SunOS UNIX (sun1)
login:
```

The `telnet` program only uses the port number of the Telnet daemon to reserve a connection. A port number can be specified optionally after the host name, when Telnet is invoked. For example, with `telnet <host name> 25`, the user can establish a connection to port 25 of the computer. This port is reserved for SMTP, which is the e-mail protocol. With the appropriate port number, the user can also establish a connection to the NNTP daemon of a newsgroup (port 119). This feature even allows the user to enter `help` to receive a description from the server daemon of its protocol commands. Such features are very useful for debugging.

- port number
- SMTP
- NNTP
- help
- MUDs and chess server
- ICS
- port 5000

In addition, on certain ports some servers offer games such as text adventures, multi-user dungeons (MUD s) (see section 12.9), and chess. The only requirements are the name of the host and the respective port number. For example, an Internet chess server (ICS), shown in Figure 10.1, is located on `anemone.daimi.aau.dk` on port number 5000. To use this server, establish a connection with `telnet anemone.daimi.aau.dk 5000`.

The program `xboard`, which normally serves as the graphical front end for the GNU chess program, is able to establish such a Telnet connection and in parallel to represent the positions graphically.

- xboard
- Internet services
- FAQs

Lists of sources for such Internet services are stored on various FTP servers under the name `internet.services` or `internet.resources`. Along with FAQ s (frequently asked questions), such lists are published regularly in the newsgroup



Figure 10.1. The ICS server

`news.answers` `news.answers`. Particularly newcomers to the Internet should learn to use a news reader, since this eases access to other servers and information sources (see section 10.8).

10.4 FTP

File Transfer Protocol

`ftpd`

`wu-ftp`

FTP is a protocol used to transfer data via the Internet. FTP stands for File Transfer Protocol and is usually implemented under Linux and other UNIX versions through the FTP server daemon `ftpd` and the client program `ftp`. Washington University's `wu-ftp` daemon is frequently used as an alternative to the simple `ftpd`. This daemon offers enhancements particularly in the areas of security and configurability.

anonymous FTP

software

Simply stated, the term FTP server usually refers to a computer offering anonymous access via FTP, rather than the server daemon `ftpd`. Large FTP servers are generally equipped with several gigabytes of hard disk capacity and offer a large selection of freeware, shareware, and public domain programs.

Linux FTP server

port 21

e-mail address

as password

To establish a connection to the most important Linux FTP server in Finland, enter `ftp nic.funet.fi`. This starts the FTP program that attempts to establish a TCP connection to the other computer via the fixed port number 21 of the FTP daemon. Once this connection has been established, the guest is requested to enter a user ID. With FTP servers that are freely accessible, guests can enter the user name `ftp` or `anonymous` and should use their own e-mail address as a password. With the commands of the `ftp` program, such as `cd`, `dir`, `get`, or `put`, the user can then search for and transfer files.

protocol

These commands do not correspond to the commands used by the FTP protocol internally, but the FTP program recognizes them and translates them into the appropriate protocol commands like `port` or `retr`. Only these protocol commands are transferred. For data transfer, a second TCP connection is established to link the data.

help

You can find individual commands on the `ftp` Manual page or in the help files of the `ftp` program, which can be retrieved by entering `help`. You will find a list of the most important FTP servers for Linux in section 5.2. The following example demonstrates an FTP session:



Figure 10.2. ftptool

```
linux2:/home/stefan>ftp sun1
Connected to sun1.
220 sun1 FTP server (SunOS 4.1) ready.
Name (sun1:stefan): ftp
331 Guest login ok, send ident as password.
Password (sun1:ftp): strobelsdemo.de
230 Guest login ok, access restrictions apply.
ftp>ls
200 PORT command successful.
150 ASCII data connection for /bin/ls (141.7.1.41,1157) (0 bytes).
total 6
drwxrwxrwx 2 0 150 512 Jul 16 16:14 Incoming
-rw-r--r-- 1 0 1 139 Aug 22 12:33 README
drwxr-xr-x 2 0 150 512 May 10 09:51 bin
drwxr-xr-x 2 0 150 512 May 10 09:54 dev
drwxr-xr-x 5 0 150 512 Jun 20 15:36 pub
drwxr-xr-x 3 0 150 512 May 10 09:52 usr
226 ASCII Transfer complete.
ftp>get README
200 PORT command successful.
150 ASCII data connection for README (141.7.1.41,1158) (139 bytes).
226 ASCII Transfer complete.
142 bytes received in 0 seconds (0.14 Kbytes/s)
ftp>bye
221 Goodbye.
```

You can also transfer files more comfortably with one of the graphical front ends for FTP rather than with the command-line-oriented `ftp` program. Many different variants of graphic front ends are available from the usual Linux FTP servers or the official server for X11 programs, `ftp.x.org`. The XView-based `ftptool` is also quite popular (see Figure 10.2).

graphical front ends

`ftp.x.org`

10.5 Archie

Given the vast number of FTP servers and the huge amount of free software available on them, it can be quite difficult to find the right software to cover a particular requirement. This is where Archie

finding programs

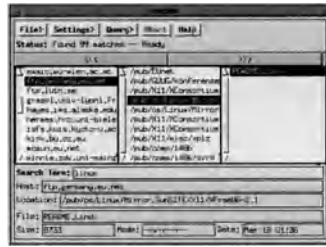


Figure 10.3. File search via xarchie

software database

servers provide valuable help. An Archie server furnishes access to a multiple-gigabyte database that contains an index of the most important FTP servers on the Internet. This database is updated automatically at regular intervals.

Archie servers are located at the following addresses:

- archie.th-darmstadt.de (Germany)
- archie.funet.fi (Finland)
- archie.ans.net (New York)
- archie.au (Australia)
- archie.doc.ic.ac.uk (Great Britain)

Telnet

To submit a query to an Archie server, log in to one of the respective hosts with `telnet` with the user name `archie` and use a simple query language to search for programs.

xarchie

The graphical front end `xarchie`, also available under Linux, provides this service with more comfort. The user needs only to enter the keyword and select the search mode, and `xarchie` establishes the connection to the already set-up server. Soon the files and their server locations are displayed in a browser. Newer versions of `xarchie` can even start an FTP transfer directly to retrieve the desired files from their respective FTP servers.

browser

whatis
description

In addition to searching for programs by name, an Archie server provides a `whatis` database in which the `whatis` command yields a concise description of a specific program. If the name of a program is not known at all or only in part, `whatis` can display a list of a relevant subset of all registered programs by name along with a description of each.

In lieu of Internet access, an Archie server can be queried per e-mail. Here the appropriate command is sent to the user `archie` on such a server. As with FTP mail servers, sending the command `help` to the Archie server in the first line of the mail produces a more detailed description.

query per e-mail

10.6 Berkeley r-Utilities

The University of California at Berkeley's BSD UNIX exerted a great deal of influence on networking capability under UNIX. Through the successful integration of TCP/IP into the UNIX operating system, this implementation made a significant contribution to the further proliferation of TCP/IP.

BSD UNIX

The Berkeley `r`-utilities are a group of programs whose names begin with the letter `r`, the most important being `rlogin`, `rsh`, and `rcp`. The `r` stands for remote. The Berkeley `r`-utilities belong to the standard software of a networked UNIX workstation. Over time they have been enriched by a number of programs such as `rwho` and `ruptime`. As in many proprietary UNIX variants, utilities from the Berkeley Distribution were ported to Linux.

`rlogin`, `rsh`, `rcp`

porting

The basic idea of these utilities is to provide a user who has an account on multiple computers on a network, a simple way to log in to other computers on the network, to run programs, or to copy files, without having to enter a password each time. To do this without creating a security flaw, certain users from other computers can be defined as *trusted users*. Only such users can then log in to the computer and use its services without having to enter a password. This definition is systemwide in the file `/etc/hosts.equiv` or in `.rhosts` in the home directories of the respective users for their own accounts.

logging in

without password

trusted users

```
#
# Accepted Hosts and Users
#
zeus.demo.de
hermes.demo.de strobelt
sun.demo.de arnold
```

Only the system administrator can modify the file `/etc/hosts.equiv`. If an individual user wants to grant access permission

`hosts.equiv`

.rhosts	to another user, this is done with an entry in the file <code>.rhosts</code> in the respective home directory. This type of file is particularly useful when
different user IDs	a user has different user IDs on different machines in the network. Appropriately set <code>.rhosts</code> files make network-wide access to a user's own directories considerably easier.
reserved port	<i>Reserved ports</i> provide an additional security measure. Here the server checks the TCP port number of the client: if it is not a privileged port, the server denies the connection. Reserved ports on
superuser privileges	UNIX machines are available only for users with superuser access privileges. This prevents a normal user with another, self-written program from feigning a false computer or user name in order to gain access to another computer. However, this means that the <code>r</code> -utilities have to run with the user ID <code>root</code> for this service.
Kerberos	Newer implementations of the Berkeley <code>r</code> -utilities also support <i>Kerberos</i> , which achieves an additional increase in security. The Kerberos system, which checks passwords and user IDs, was developed at the Massachusetts Institute of Technology (MIT).

rlogin

Telnet	For the user, the remote login program <code>rlogin</code> functions similarly to the <code>telnet</code> program, except that with an appropriate configuration it requires no user ID or password. <code>rlogin</code> does not permit the specification of a deviating port number.
--------	--

```
hermes:/home/strobel> rlogin zeus
zeus:/home/strobel>
```

rcp

access path	<code>rcp</code> supports remote copying of files between computers. However, the exact access path must be specified. Therefore, in many cases users prefer the interactive <code>ftp</code> utility or access via a Network File
NFS	System (NFS). In contrast to normal FTP, <code>rcp</code> can also recursively copy subtrees of file hierarchies.

```
zeus:/home/strobel> ls
Amster.txt  README      hn-net.tki  lainel/     maurer.txt
Buch/       emacsst.txt kernel.txt  mail/       mib.txt
```

```
zeus:/home/strobel> rcp -r strobel@sun1:/home/prog/xprogs
zeus:/home/strobel> ls
Amster.txt  README      hn-net.tki  lainel/     maurer.txt
Buch/       emacsst.txt kernel.txt  mail/       mib.txt
xprogs/
```

rsh

The remote shell `rsh` allows a user to execute programs on other computers. Beyond the command to be executed, the name of the remote computer and, optionally, a user ID are specified. The output of the executed command is then routed back to the local machine via the network.

Beyond the remote execution of commands on other machines, this program is also adept at fast data transfer between computers. Here the user exploits the `rsh` feature that allows combining the local machine's own standard input and output with the program that it executes on the target computer. For example, this enables making a backup of files from a local hard disk onto a streamer on another computer on the network.

programs

data transfer with rsh

backup files

```
stef1:/home/strobel> rsh lia "ls .em*"
.emacs
.emacs-bkmrks
.emacs-places
.emacs-places
.emacs-skp
.emacs
stef1:/home/strobel>
stef1:/home/strobel> rsh lia "tar cfz - .em*" | tar xvfz -
.emacs
.emacs-bkmrks
.emacs-places
.emacs-places
.emacs-skp
.emacs
stef1:/home/strobel>
```

10.7 Mail

The TCP/IP e-mail service usually consists of a daemon, which transfers messages to other computers with the SMTP protocol, and programs for reading and writing mails. These are also called mail readers.

Under Linux, both the popular `sendmail` program and the alternative `smail` are available for transferring mails. Aside from transport via a TCP/IP network, these also work well for UUCP.

daemon

mail readers

sendmail and smail

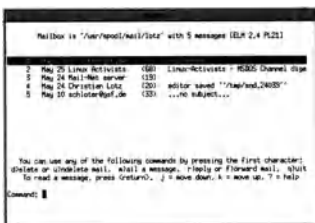


Figure 10.4. elm

Most packets contain `pine` and `elm` as mail readers. In addition, there are also graphic programs such as `mumail`, which allow for easy mail administration under X11.

elm

older, but widely
propagated

`elm` is a somewhat older, but widely propagated program for reading and writing e-mail (see Figure 10.4). It exists for most UNIX platforms. When `elm` is first started up, two subdirectories are created in the user's home directory: a configuration directory and a directory for storing received mails. A list displays new mail with the sender and the title of the message. The mail can be selected with the cursor keys, then read and stored as files.

external editor

To write mails, the user defines an external editor in the `elm` options. `elm` also handles mails in Multipurpose Internet Mail Extension (MIME) format, which can contain graphics, sounds, or programs. These MIME mails can be output with the `metamail` program and the respective presentation programs.

Pine

Internet News

Pine, which stands for “Pine Is No longer Elm” or Program for Internet News & E-mail, affords more comfort than `elm`. One of its major differences with respect to `elm` is `pine`’s feature allowing access to the Internet News (see section 10.8). `Pine` contains an editor called `pico` that significantly eases the production of mail. Since `pine` uses the IMAP2 protocol, it can also manage mailboxes that are located on a remote machine. This proves particularly interesting for a user who often switches among several machines.

IMAP

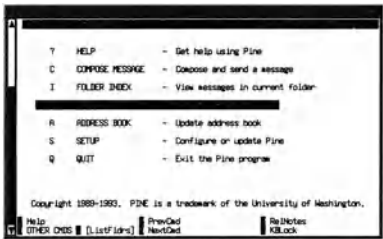


Figure 10.5. The mail front end Pine

IMAP2 assures consistent access to the mail of the home machine and now supports the MIME standard as well.

Use the file `.pinerc` in the user's home directory to configure Pine. Settings for all users can be stored in the file `pine.conf`, which is normally located in the directory `/usr/local/lib`. Below are some of the most interesting settings:

`.pinerc`
`pine.conf`

```
user-domain=demo.de
```

This defines the domain name in outgoing mails. If someone with the user name `mueller`, for example, at the computer `hermes.demo.de` sends a mail, then the sender address is entered as `mueller@demo.de` according to the setting shown above.

sender address

```
inbox-path=mail/inbox
```

The setting `inbox-path` determines the path of the folder for incoming mails. This folder is usually a file with the user's name in the directory `/var/spool/mail`. Inside Pine, the folder has the name `INBOX`. If mail arrives in a different directory, or if a program such as `deliver` automatically sorts the messages into different files, the file path can be changed.

path for incoming

`/var/spool/mail`

```
incoming-folders=Linux mail/linux,  
Projekt mail/project,  
Zeus {zeus.demo.de}
```

additional mailboxes

title and path

IMAP

feature-list

old growth

Additional mailboxes can be entered in the `incoming folders` line. These folders may be other local files or mailboxes on other computers. A title and a path are defined for every folder. If a mailbox is on another computer, the name of the computer is given in brackets before the path. The mailboxes are accessible in this case via the IMAP2 protocol.

With the variable `feature-list` you can influence `pine`'s behavior. Possible settings are described in the commentaries of the `.pinerc` file. Frequently used settings are `old-growth` and `auto-move-read-msgs`. The latter automatically moves mail that has been read into the folder called `read-messages` when `pine` is closed.

```
feature-list=old-growth, auto-move-read-msgs
```

graphics You can also define the program that `pine` invokes to view graphics in MIME mails.

```
image-viewer=xv
```

MuMail

X11 One program with a graphic interface is `mumail` (see Figure 10.6). This program uses X11 and can be comfortably used with the mouse. The range of its functions is approximately the same as `elm`. MuMail can also be used to create and view multimedia mail.

deliver

sorting mail There are several programs available for automatically sorting mail, `deliver` and `procmail` being among the most widespread. We use `deliver` as an example to show how this kind of program works and how to configure it.

incoming mail First of all, `deliver` must have control over incoming mail. A user who wishes to have mails sorted can pass them on to `deliver` with an entry in the file `.forward`. The contents of this file then looks like this:



Figure 10.6. Mail front end MuMail

```
|/usr/bin/deliver benutzername
```

As an alternative, you can integrate `deliver` directly into the configuration of the mail transport daemon transport. This is described in section 10.7 under the configuration of `smail`.

Following this, `deliver` is invoked with every arriving mail, and the message is transferred to `deliver`'s standard entry. `deliver` writes the header and the contents of the mail in a temporary file and invokes a script called `.deliver` in the user's home directory. This script determines the mail's target. The individual user must create the script, and it must be able to analyze the contents of the mail. The analysis is usually based on keywords in the sender address or the subject line.

The script outputs to the standard output what should happen with the message. The syntax of the most important output possibilities is listed below.

- `:Mailbox` — Appends the current mail to the file listed.
- `|Command` — Invokes the defined command and transfers the mail to the standard input.

smail

standard output

The following example demonstrates such a script:

```
user="$1"
KENN='/usr/bin/header -f To -f CC -f SENDER -f Reply-To $HEADER'
SUBJECT='/usr/bin/header -f Subject $HEADER'

# Filter Mailinglisten
case "$KENN" in
  *tkined@ibr.cs.tu-bs.de* )           echo :Mail/IN/tkined;    exit
  ;;
  *samba@* )                          echo :Mail/IN/samba;    exit
  ;;
  *new-tty* )                         echo :Mail/IN/linux;    exit
  ;;
  *linux* | *Linux* )                 echo :Mail/IN/linux;    exit
  ;;
  *Firewalls-Digest* | *firewalls-digest*) echo :Mail/IN/firewall; exit
  ;;
esac

case "$SUBJECT" in
  *Daily*" Usenet*" report*)          echo :Mail/IN/news;    exit
  ;;
  *Quota*" usage*" on*)              echo :Mail/IN/system;  exit
  ;;
esac

# Andere Mails
echo ":Mail/IN/default"
```

central scripts The system administrator can install `deliver` to invoke a central script before the `.deliver` scripts, so that mails are delivered to the various users first. You will find more details on the Manual page for `deliver`.

mail configuration

transport There are various programs available for simply transport ing mails.

mail Some distributions use the program `sendmail`, but most use `smail`. In this section we describe the configuration of `smail` for a computer with a direct Internet connection connection. The modifications for a UUCP connection are explained subsequently.

UUCP

/var/lib/smail The files in the directory `/var/lib/smail` contain the configuration of the `smail` daemon. Some distributions combine the configuration files with the help programs in the directory

compilation `/usr/lib/smail`. `smail` can be customized on compilation or later in the configuration files. In the latter case, the settings in the file overwrite the values defined on compilation, which are normally unusable.

The most important configuration files files are

- `config`
- `routers`

- transports
- directors

The `config` file contains global settings such as the name of the computer. The entries are normally in the form `attribute=value`. The `#` character introduces a comment.

global settings

```
#
# smail Konfiguration for hermes (our mail server)
#
hostname=hermes.demo.de
more_hostnames=demo.de
```

`hostname` defines the name of the local computer. It is used in the header of outgoing messages and also for recognizing incoming messages to the local computer. `more_hostnames` lists alternative names. In the example above, `hermes` is the mail server for the domain `demo.de`. The entry `more_hostnames=demo.de` effects that the actual computer name can be left out of mail addresses. Thus `strobел@demo.de` is sufficient, rather than the long address `strobел@hermes.demo.de`.

`more_hostnames`

The files `routers`, `transports`, and `directors` have more complicated tasks. They use the address to determine the path and the type of delivery. First the address is separated into a *target address* and a *remainder*. In the address `strobел@demo.de`, the target address is `demo.de` and `strobел` is the remainder. The target address provides information about whether the mail is local or whether it needs to be forwarded to another computer. For local mail, the file `directors` helps to determine how the mail should be delivered to the addressee. The file `routers` determines the route for remote addresses and defines the protocol to be used. The options of the various types of transport (SMTP via TCP/IP, UUCP, etc.) are defined in the `transports` file.

`routers`, `transports`
and `directors`

target address

local?

route

The file `routers` contains a list of router entries. These routers have nothing to do with IP routers. They define an instance, which has a driver assigned to it, within the `smail` program. `smail` forwards the mail addresses one after the other to the defined routers, which then check whether they can process each address.

`routers`

driver

attributes

characteristics

Generic and driver-specific attributes are defined for every router in the file `routers`. The generic attributes can be entered for each computer and they define the driver and the transport to be used. The specific attributes can influence the individual characteristics of the particular drivers.

The `routers` file in the following example contains definitions for a computer with both an Internet connection and UUCP connections. Since entries for MX records resolution are missing in many distributions, `smail` cannot be used for Internet without reconfiguration.

```
#
force_paths:
    driver=pathalias,
    transport=uux,                # Use the uux transport
    always;                      #
    file=forcepaths,             # Name of the file
    proto=lsearch,              # direct access (no dbm files)
    optional;                   # The file is optional

#
match_inet_addr:
    driver=gethostbyaddr,        # processes IP addresses in {}
    transport=smtp;             # Transport is smtp
    fail_if_error;
    check_for_local

#
match_mx_hosts:
    driver=bind,                 # Resolution of MX records
    transport=smtp;             # TCP/IP SMTP
    defnames;                   #
    defer_no_connect;           # retry if name server is down
    local_mx_okay;

#
match_inet_hosts:
    driver=gethostbyname,        # resolve host names with resolver
    transport=smtp;
    domain =

#
smart_host:
    driver=smarthost,            # special-case driver
    transport=smtp;             # by default deliver over SMTP
```

Internet and UUCP

The first router entered here is only necessary if UUCP connections are used in addition to an Internet connection. The router enables the creation of a file called `forcepaths`, which contains the domain names and target addresses for `uucp`. Mail to domains that are entered in this file is sent to the target address entered in this file.

IP addresses

The router `match_inet_addr` makes it possible to use IP addresses in a mail address. If it is not possible to resolve a symbolic name for any reason, you could use the address `strobela@[141.7.41]` instead of `strobela@hermes.demo.de`.

`match_mx_hosts` is the router normally used for mail on the Internet. It resolves the MX entry for a symbolic address.

The `match-inet-hosts` entry can resolve addresses with help from the resolver (see section 9.8). This enables the use of symbolic addresses in the form `name@computeraddress`. In this case, however, MX records cannot be resolved. Thus a mail could be delivered to `strobел@hermes.demo.de`, but not to `strobел@demo.de`, because there is no computer with this name. There is only an MX record that refers to `hermes`.

The `smarthost` router forwards all mail to another computer. This router is used if there is a central mail gateway that is responsible for mail delivery, and the local computer is not intended or able to deliver mail directly. The address of the mail gateway to which the `smarthost` driver should forward mail is usually defined with the variables `smart_path` in the `config` file. The transport (see below) can also be entered with the variable `smart_transport` in the `config` file. These settings overwrite the attributes that may have been entered in the file `routers`.

We discuss the use of the `smarthost` router further in conjunction with the `uucp` settings.

The characteristics of the individual transport possibilities are defined in the file `transports`. This has the same structure as the file `routers`. Both generic and driver specific attributes can be entered for every transport.

MX records

resolver

smarthost
gatewaysmart_path
smart_transport

transports

```
local: driver = appendfile,          # append message to a file
      return_path,                 # include a Return-Path: field
      local,                       # use local forms for delivery
      from,                        # supply a From_envelope line
      unix_from_hack;              # insert > before From in body
      file = /usr/spool/mail/${lc:user},
      group = mail,                # group to own file
      mode = 0660,                 # group can access
      suffix = "\n",               # append an extra newline
      append_as_user,

#
pipe: driver = pipe,                # pipe message to another program
      return_path, local, from, unix_from_hack;
      cmd = "/bin/sh -c $user,"    # send address to the Bourne Shell
      parent_env,                  # environment info from parent addr
      pipe_as_user,                # use user-id associated with address
      umask = 0022,                # umask for child process
      -log_output,                 # do not log stdout/stderr
      ignore_status,               # exit status may be bogus, ignore it
      ignore_write_errors,         # ignore broken pipes

#
file: driver = appendfile,
      return_path, local, from, unix_from_hack;
```

```

file = $user,                                # file is taken from address
append_as_user,                             # use user id associated with address
expand_user,                               # expand    and $ within address
suffix = ".n,"
mode = 0644

#
uux:    driver = pipe,
        uucp,                                # use UUCP-style addressing forms
        from,                               # supply a From_envelope line
        max_addr = 5,                       # at most 5 addresses per invocation
        max_chars = 200,                    # at most 200 chars of addresses
        cmd = "/usr/bin/uux -r gSgrade $host!rmail ${!$(strip:user)!},"
        umask = 0022,
        pipe_as_sender

#
smtp:   driver = smtp,
        max_addr,
        max_chars

```

local mails The transport `local` delivers mail to local users. It appends
/var/spool/mail the text to the mailbox file of the appropriate user in the directory
attributes `/var/spool/mail`. The driver used for this is called `appendfile`.
Its specific attributes define the user, the group and the access
permissions with which the mailbox file is to be written.

pipe The transport `pipe` is used when mail is to be forwarded to a
program with a pipe upon delivery to the local computer. This may
.forward be the case if users have created a file `.forward` in their home
directory, which contains a pipe with a program name as an address,
for instance `|/usr/bin/deliver user` (see above).

file `smail` uses `file` implicitly if an address contains a path name.
As with `local`, it uses the `appendfile` driver.

-r The transport `uux` delivers mails via `uucp`. As with the transport
pipe, it uses the driver `pipe` to forward mail to a program's standard
output. Using the `-r` flag in the invocation of `uux` prevents immediate
delivery of mail. In this case, they are stored in the `uucp` spool
directory and forwarded at the next poll.

directors The process of delivering local mail is defined in the file
aliases `directors`. The entries in this file primarily direct internal features,
such as the expansion of addresses from the file `/usr/lib/aliases`
or forwarding mail via the file `.forward`. It has the same format as
the files `routers` and `transports`.

```

aliasinclude:
driver = aliasinclude,
nobody; copysecure,
copyowners,

forwardinclude:
driver = forwardinclude,    # use this special-case driver

```

```

nobody;
copysecure,          # get perms from forwarding director
copyowners,          # get owners from forwarding director

aliases:
driver = aliasfile,   # general-purpose aliasing director
nobody,              # problems go to an owner address
owner = owner-$user;
file = /usr/lib/aliases,
modemask = 002,
#proto = dbm,         # use dbm(3X) library for access
proto = lsearch,     # use linear search through text file

forward:
driver = aliasfile,   # general-purpose aliasing director
nobody,              # problems go to an owner address
owner = real-$user;
file = /var/lib/aliases/forward,
modemask = 002,
proto = lsearch,

dotforward:
driver = forwardfile, # general-purpose forwarding director
owner = Postmaster,  # problems go to the site mail admin
nobody,
sender_okay;         # sender never removed from expansion
file = /.forward,    # .forward file in home directories
checkowner,          # the user can own this file
owners = root,       # or root can own the file
modemask = 002,       # it should not be globally writable
caution = daemon:root, # don't run things as root or daemon
unsecure = "uucp:uuucp:/tmp:/usr/tmp"

user: driver = user;   # driver to match usernames
transport = local     # local transport goes to mailboxes

lists: driver = forwardfile,
caution,             # flag all addresses with caution
nobody,              # and then associate the nobody user
owner = owner-$user;
file = lists/${lc:user} # lists is under $mail_lib_dir

owners: driver = forwardfile,
caution,             # flag all addresses with caution
nobody,              # and then associate the nobody user
owner = postmaster;
prefix = "owner-,"
file = lists/owner/${lc:user} # lists is under $mail_lib_dir

request: driver = forwardfile,
caution,             # flag all addresses with caution
nobody,              # and then associate the nobody user
owner = postmaster;
suffix = "-request,"
file = lists/request/${lc:user} # lists is under $mail_lib_dir

```

The entries under `lists`, `owners`, and `requests` are especially interesting. They make it possible to create mailing lists very simply. Here it is sufficient to create a file in the directory `/var/lib/mail`, which uses the address of the list as a name. The addresses of all list recipients are entered in this file.

`lists`
mailing lists
address

mail and deliver

If there are a number of users who want their mail sorted into different mailboxes by a program such as `procmail` or `deliver` (see above), we recommend integrating the program directly into the transport

`deliver`
transport local

`local`. Using the example of `deliver`, we will demonstrate in this section how the necessary changes should be made.

`pipe` The transport `local` uses the driver `pipe` instead of the driver `appendfile`. Use the complete path of `deliver` as the program. Normally this is `/usr/bin/deliver`.

```
local: driver = pipe,
      return_path,
      local,
      from,
      unix_fromHack;
      cmd = "/usr/bin/deliver ${lc:user},"
      parent_env,
addr   pipe_as_user,
address umask = 0022,
      .ignore_status,
      .ignore_write_errors,
```

`.deliver` Users wishing to have their mail automatically sorted into various folders then only need to enter a script with the name `.deliver` into their home directories.

smart and UUCP

`smarthost` If you do not have a direct Internet connection or want to transport mail only via UUCP for other reasons, then the simplest method is to use the `smarthost` router and forward all outgoing mails to a computer with a better connection. To do this, remove all entries except `smarthost` from the file `routers` and set the variable `smart_path` to the address of the mail gateway and the variable `smart_transport` to `uux` in the `config` file file.

You will find more detailed information and the configuration details on the Manual page for `smart`.

10.8 News

`information` The Internet News is one of the most important sources of information about new developments and many other subjects.

Structure and makeup

`news server` The principle is quite simple. News servers have been set up at many locations around the Internet, for example, at almost every university.

A news server manages various news groups, which amount to bidirectional electronic bulletin boards, each on a certain subject area, for posting, reading, and replying. These subjects range from operating systems and programs to sports, computer games and other recreational activities, to matters of social interest. Several thousand such newsgroups exist worldwide.

many topics

several thousand
groups

All news servers are constantly exchanging postings with neighboring news servers, so that a message that is posted on a given news server in a certain group quickly propagates to all news servers. These postings remain on the news servers for a certain time, for example, two weeks, before they are deleted.

The entirety of all computers that exchange these postings in the form of news is called Usenet. Communication between these machines was originally based on modems and UUCP (UNIX to UNIX copy). This made it significantly slower than the current connections on the Internet, which meanwhile uses a special protocol called NNTP instead of UUCP.

Usenet

UUCP

NNTP

Newsgroups

The newsgroups are structured hierarchically, as shown in Figure 10.7. Their full names, separated by periods, reflect their position in this hierarchy. `comp.os.linux.announce`, for example, means that this newsgroup resides under the heading `comp`, which deals with computers, software, and computer science in general. `os` indicates operating systems, and `comp.os.linux` is the leader for all names of groups under the `comp` hierarchy that have to do with Linux.

newsgroup names

`comp`

Linux

In addition to `comp` there are many other important hierarchies such as `sci` for groups that discuss scientific subject matter and `alt` for a broad range of alternative subject areas.

`sci`

Moderated groups

Groups can be either open to any NetNews participant or moderated. In the latter case a moderator decides which proposed postings might be of interest to the readership of the newsgroup; these are posted, while the rest find their way into a black hole. For example, a moderated newsgroup for Linux is `comp.os.linux.announce`, which posts exclusively announcements of new programs or system

moderator

new programs

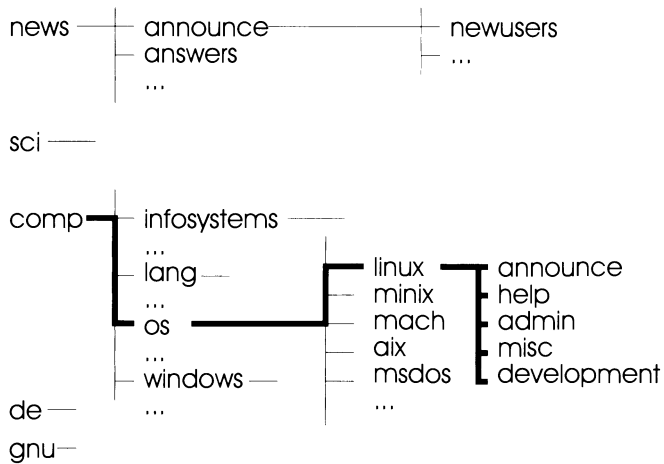


Figure 10.7. Excerpt from the newsgroup hierarchy

extensions. This is especially necessary because the number of postings in the other Linux groups is so overwhelming that a participant has to invest a great deal of time to read them and keep up to date.

open groups Unmoderated groups are open, so that every participant can post statements, questions and replies.

Rules and netiquette

A new newsgroup is created if it is recommended by a Usenet participant and the suggestion is accepted in a network vote.

discussions Discussions about proposed new groups usually take place in the
new groups group `news.groups`.

rules When a participant wants to post a message to a newsgroup, certain rules should be observed, the network etiquette, or netiquette. Insulting comments or personal attacks are not tolerated. The newsgroup `news.announce.newusers` presents an overview of these rules of network behavior.

News reader

Reading the NetNews requires a news reader and a news server that is accessible via network or modem. At universities this is



Figure 10.8. xvnews newsreader

usually no problem since they generally operate their own news server.

News readers have become available for almost all computer platforms and operating systems that support a network (see Figure 10.8). Well-known news readers include `rn`, `trn`, `tin`, `xrn`, and `xvnews` for the X Window System and `trumpet` for PCs. Almost all of these news readers exist under Linux. The only configuration required for most news readers is an environment variable named `NNTPSERVER`, which must contain the name of the news server (see section 2.7).

Another workable option is `xrn`, which is available in an Athena widget version and a Motif variant (see section 8.5).

News server

Installing a news server is somewhat more complicated than installing a client. For a news reader, it suffices to copy the program to the computer and set an environment variable. A server, however, needs to have several programs and configuration files installed. In order to receive news, you need another news server that forwards the news to you.

available options

tin, xrn, xvnews

NNTPSERVER

xrn

installation

several programs
and config files



Figure 10.9. `xrn` under OSF/Motif

Internet usually uses the server INN in addition to the older C news server program C news. The configuration of INN is described in a text file in the `tar` file of the server source code.

10.9 IRC

Internet Relay Chat	Internet Relay Chat (IRC) offers a means to converse directly with several participants on the Internet. Similar to NetNews, IRC represents a system of networked servers that exchange information among themselves. As with NetNews there is a hierarchical structure according to subjects, called <i>channels</i> under IRC (see Figure 10.10).
---------------------	---



Figure 10.10. IRC

What distinguishes IRC from NetNews is that the exchange of messages occurs without delay and the user can converse directly with the other participants. This proves particularly interesting when there is an acute problem that needs to be discussed with others.

real-time conversations

The user gains access with a special `irc client` client program that establishes the connection to the next IRC server. The IRC client has certain commands that all begin with a slash (/). These commands enable the user, for example, to sign on to and sign off from a channel. Text that is entered without a command is immediately transferred and is displayed for all other IRC participants that are currently signed on to the same channel.

`irc client`

`/ commands`

IRC also has a channel that is dedicated to Linux discussions. To take part in this discussion, start the IRC program and enter `/join #linux`. `/join` is the command to sign on to a channel. Then every message that is posted on the channel is displayed along with the name of the sender.

Linux channel

The number of channels changes constantly, since every user can open new channels. This is also done with the `/join` command. If the specified channel does not exist, it is created. Normally hundreds of channel are active simultaneously; all currently active channels can be displayed with the `/list` command.

new channels

`/list`

Besides its on-line discussion feature, IRC protocol also permits the exchange of smaller files. Furthermore, private messages can also be transmitted.

exchange of files

Participating in IRC is also possible and considerably more convenient with the graphical client `zircon` (see Figure 10.11). Zircon is operated entirely with the mouse and displays every open channel in a separate window.

`zircon`

`Zircon` is written with Tcl/Tk and requires an enhanced Tcl interpreter named `tcl-dp` to access TCP/IP and sockets (see section 14.5).

`tcl/dp`

10.10 Gopher

Gopher is an Internet service that makes available many other services provided by universities and other institutions worldwide and combines them under a single user interface. Gopher presents itself to the user as a single, large, hierarchical menu. Within this

single user interface



Figure 10.11. The graphic IRC front end zircon

structure the user can move from one server to another and employ databases various network services. The offer includes literature databases, weather services, all kinds of documents (text, pictures, sound), cafeteria menus at various universities, Telnet sessions, and gateways menu structure to other services such as WAIS. An increasing number of institutions are providing information via their Gopher service that previously was available only in printed form.

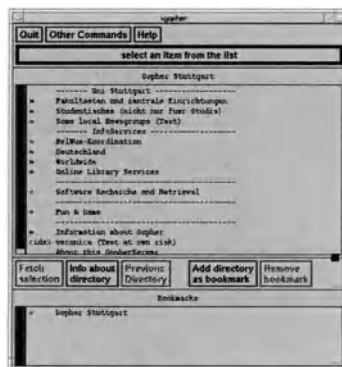


Figure 10.12. xgopher

To be able to use Gopher, the user needs a Gopher client. Under Linux this could be the `xgopher` that runs under X11. When it is started, `xgopher` establishes a connection to a Gopher server, and the user can move freely in the Internet (see Figure 10.12).

The significance of Gopher has greatly decreased in the past few years with the growing popularity of the World Wide Web. Almost all of the institutions that originally offered information via Gopher have now switched over to WWW.

decreasing significance

10.11 World Wide Web

World Wide Web (also referred to as WWW or W3) is an Internet service that employs distributed multimedia hypertext documents that can contain not only formatted texts but also references to other documents, pictures, videos, or sound files. These documents are described in the HyperText Markup Language (HTML) in ASCII format. A special protocol called HyperText Transfer Protocol (HTTP) is used for data transfer.

WWW

documents, pictures,
videos and sound
HTML
HTTP

References in HTML documents are specified with Uniform Resource Locators (URL s), which include the `protocol://host address:port/path`. Specification of the port number is optional; if no port is specified, the standard port for the specified protocol is used. Such URLs permit the expression of references to other HTML files as well as almost all kinds of addresses of services on the Internet. For example, the URL for the directory `/pub` on the FTP server `tsx-11.mit.edu` is `ftp://tsx-11.mit.edu /pub`.

URL
port number

all types of addresses

In order to access WWW, you need a WWW client. These are also referred to as WWW browsers. The first and best-known of these is `Mosaic`. It is available for all common operating systems and is free of charge for private use. Newer commercial PC operating systems such as Windows 95 and OS/2 Warp already contain WWW clients. Translating `Mosaic` requires `OSF/Motif`, which is not freeware (see Figure 10.13). However, there is also an already-compiled version that is statically linked and thus does not need the `OSF/Motif` run-time system.

WWW browser
Mosaic

Motif

In addition to `Mosaic`, newer WWW browsers such as `Netscape` and `Arena` are also available for Linux. `Netscape` is actually a

Netscape

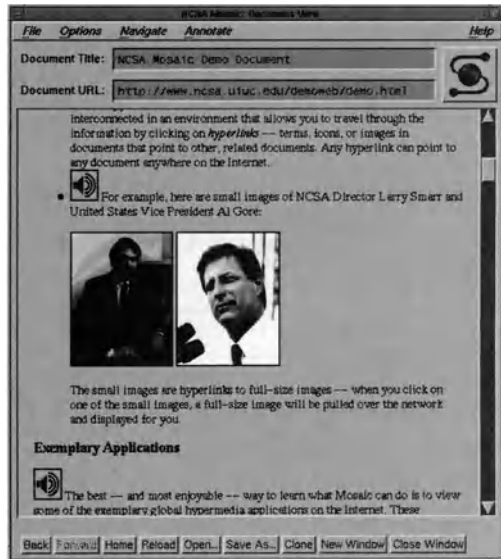


Figure 10.13. WWW access via Mosaic

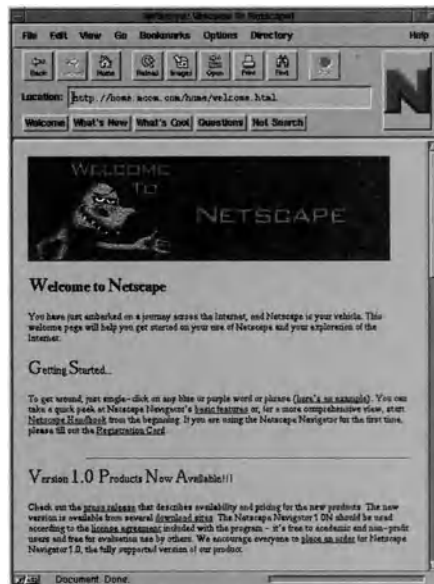


Figure 10.14. The WWW client Netscape

commercial product, which includes the advantage of its treatment of security. A transfer can be RSA-encoded in conjunction with special servers. This makes Netscape interesting for commercial applications that transfer private data such as credit card information.

RSA
credit card data

Arena is the first implementation of a browser using the new HTML3 standard, which should replace the HTML used until now for describing WWW sites in the future. This development is still in process, which means that Arena is primarily an area of academic interest at the moment.

HTML3

With the help of an HTTP server, a distributed information system can be constructed easily under Linux. Such a server, running in the background as a daemon, waits for a socket connection via a defined port; after a connection has been established by a WWW client (e.g., Mosaic), the server furnishes access to the files in certain directories. If direct Internet access is possible, the local documents can offer appropriate links to other HTTP servers. Interesting servers for Linux users can be found at the following URLs:

HTTP Server
daemon
links to other servers

<http://www.linux.org/> WWW server with an extensive list of FAQs, HOWTOs, manuals, and other WWW servers.

<http://www-i2.informatik.rwth-aachen.de/Linux.html> WWW server of the Linux archive at the RWTH Aachen (Germany). It contains information on Linux, access to other FTP servers, and other WWW servers.

<http://www.linux.org.uk/> This British Server is managed by Alan Cox. It contains a list of commercial software for Linux.

10.12 Network management

tkined is a particularly interesting tool for the area of network management (see Figure 10.15). It allows the supervision and administration of local TCP/IP networks and WAN s. Its functions include automatic recognition of a network structure, appealing graphical representation, monitoring connections and individual components, administration of SNMP-enabled elements, and many others. Since tkined was developed with Tcl/Tk, it is relatively easy to enhance it and make it suitable for particular needs. It uses an enhanced Tcl interpreter named scotty, which enables access

TCP/IP networks
monitoring
scotty

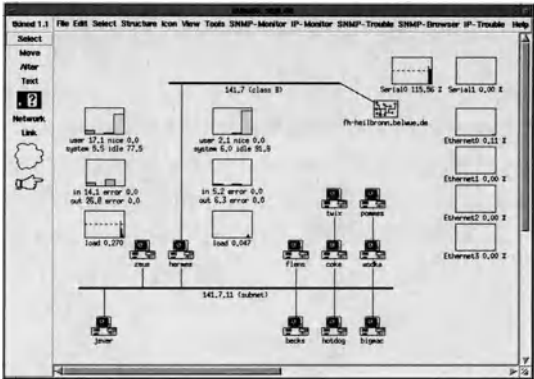


Figure 10.15. tknet

to the protocols TCP, UDP, ICMP, DNS and SNMP under Tcl (see section 14.5).

Support & Help

One argument that is frequently posed, especially in commercial settings, to reject free or public domain software is the lack of a licensing company that provides support and a hotline. In the USA some companies have recognized this market niche and offer commercial support for free software. A similar trend is developing in Europe as well (see Appendix).

hotline

free software

Apart from such support, there are many possible ways to receive information and help via the Internet or directly from the Linux system for concrete problems. This chapter offers an overview of these possibilities and of the available documentation for Linux.

Internet

11.1 man, xman

As with every UNIX system, a simple source of information is the on-line documentation. These are files that each describe one command, one C library routine, a device, or the contents of a configuration file. The command for displaying these On-Line Manual pages is `man`. For this reason, these files are usually called Manual pages, or simply `man` pages.

on-line documentation

The On-Line Manual pages are divided into sections according to contents. These sections are labeled with the numbers 1 to 9 and the letters n and l as follows:

- 1 - Commands for the user
- 2 - System invocations
- 3 - Routines of the C library
- 4 - Device-files in `/dev`
- 5 - File formats (usually configuration files)

- 6 - Games
- 7 - Miscellany
- 8 - System administration
- 9 - Kernel routines
- n - New Manual pages
- 1 - Local Manual pages

directories You can find the Manual pages in various directories, where the individual files of a section are stored in subdirectories. Usually the directories `/usr/man` and `/usr/local/man` are used. The example below shows the subdirectories for the sections in the directory `/usr/man`.

```
hermes:/usr/man# ls
cat1/   cat6/   de/      lpman    man4/    man8/    mann/
cat2/   cat7/   doman*   man1/    man5/    man9/    nl/
cat4/   cat8/   fix.so*  man2/    man6/    manX/    whatis
cat5/   catX/   german/  man3/    man7/    man1/
```

Other directories or subdirectories can be used as well, depending on the version and the configuration of the `man` command.

administrator The system administrator can use the command `makewhatis` to create an index file of the Manual pages from the brief descriptions contained in them. The index file contains only the name of the commands and a one-line description. This file already exists in most of the common Linux distributions. The user can use the commands `whatis` and `apropos` to search for key words.

```
hermes:/usr/man# whatis gcc
gcc (1)          - GNU project C and C++ Compiler (v2.4)
hermes:/usr/man# whatis groff
groff (1)        - front end for the groff document formatting system
groff (1)        - formatiert Texte (z.B. Manualpages)
hermes:/usr/man# apropos gif
gif2tiff (1)     - create a file from a GIF87 format image file
giftopnm (1)     - convert a GIF file into a portable anymap
giftorle (1)     - Convert GIF images to RLE format
pmtogif (1)      - convert a portable pixmap into a GIF file
rletogif (1)     - Convert RLE files to GIF format.
hermes:/usr/man#
```

Format of the On-Line Reference Manual pages

The On-Line Manual pages are sometimes available in two or three different formats. The original format is usually source code for

`nroff` and is located in the directories `man1` to `man8`. This format is unsuitable for direct screen output and so must be translated by the text-formatting program `nroff` or `groff` (see section 12.4).

For more complex On-Line Manual pages, this procedure takes some time; therefore, after the files have been translated once, they are stored in readable form in the directories `cat1` to `cat8`. To do this, however, the user must have write permissions for these directories. If desired, the files can also be compressed with `compress`, which only insignificantly increases the time required for displaying them but radically reduces disk storage consumption.

If additional On-Line Manual pages have been installed in a directory other than `/usr/man` (e.g., `/usr/local/man`), the path for the On-Line Manual pages can be defined with the environment variable `MANPATH`. The following defines the path for the On-Line Manual pages in a Bourne shell:

`nroff``groff``cat1` to `catn``compress``MANPATH`

```
export MANPATH=/usr/man:/usr/openwin/man:/usr/local/man
```

This example defines the path for the On-Line Manual pages in a C shell:

```
setenv MANPATH /usr/man:/usr/openwin/man:/usr/local/man
```

Naturally, this setting should be made in the appropriate startup file (`.profile` or `.login`).

xman

The X11 affords the somewhat more comfortable utility `xman` (see Figure 11.1). Here the user first selects a section of the Manual. Then the utility provides an overview of all On-Line Manual pages in that section. With the mouse the user can select an On-Line Manual page from this overview for display.

X11

When a problem arises with a command or with the configuration of a program, the first point of reference should always be the respective On-Line Manual page. Naturally the `man` command also

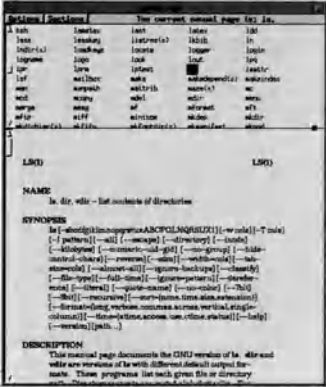


Figure 11.1. xman

has its corresponding On-Line Manual page, which the user invokes with `man man`.

11.2 Info

GNU	The documentation of many FSF programs is available in GNU Info format. GNU Emacs uses this format to provide hypertext-like navigation through these documents. Newer versions of the Emacs
Emacs 19	editor, such as Lucid Emacs and Emacs 19, permit controlling the navigation with the mouse under the X11. There are also programs
tkinfo	like <code>xinfo</code> and the TCL/Tk-based <code>tkinfo</code> that were conceived exclusively for displaying info files (see Figure 11.2).

Emacs Within the Emacs editor the user invokes the info mode (**<Strg-H><I>**) and selects from the menu of available info documents the needed text. In this text the user can navigate hierarchically from keyword to keyword.

Examples of Info documents include the documentation to the GNU C compiler and the GNU AWK utility.

11.3 Newsgroups

Since Linux continues to be developed on the Internet, the many communication services of the Internet are utilized to the fullest.



Figure 11.2. tkinfo

Among these services, the newsgroups provide an information source of particular interest to Linux novices (see section 10.8). Linux enjoys several newsgroups that provide a forum for discussion of questions about the system and its installation as well as other subjects. The most important newsgroups are the following:

Linux groups

- comp.os.linux.announce (c.o.l.a.)
- comp.os.linux.help comp.os.linux.misc
- comp.os.linux.admin
- comp.os.linux.development.apps
- comp.os.linux.development.system
- comp.os.linux.hardware
- comp.os.linux.networking
- comp.os.linux.setup
- comp.os.linux.x
- comp.os.linux.advocacy

comp.os.linux.announce .announce is a moderated newsgroup for the announcement of new programs or ports to Linux. Messages intended for posting in a moderated group are not posted directly, but are first filtered by a moderator who is responsible for maintaining the respective rules within the newsgroup. The other four groups have no restrictions. Any participant can post messages or questions here. However, users should be aware that every message that is posted to a newsgroup is propagated around the world and consumes storage on every news server.

moderated

propagated
around the world

shared problems
and solutions

Currently over 100 new postings appear in these newsgroups daily. This makes it difficult to keep up to date without perusing these postings more often than on a weekly basis. A newsgroup observer can often solve problems simply by reading the newsgroup postings and following the problems and solutions that other users share.

11.4 FAQs and HOWTOs

frequently asked
questions

Another important information source that is closely linked to the newsgroups is the FAQ (frequently asked questions). This is a list of questions that arise often in postings and, more important, the answers to these questions.

questions & answers

FAQs cover a broad range of subjects. They are usually compiled by active participants of the newsgroup, with the intention of avoiding the repeated posting of the same questions. A FAQ thus consists of a series of such questions and the corresponding responses provided by various competent newsgroup participants.

HOWTO

Some of the original Linux FAQs have evolved into HOWTO s. These documents resemble FAQs but contain more detailed text.

text files

Both FAQs and HOWTOs are plain text files, so that they can be read and printed with any text editor, with the UNIX `more` command, or even with DOS. As a rule, Linux FAQs and HOWTOs do not cover general UNIX questions. A dedicated newsgroup exists for this purpose (`comp.unix.questions`) with its own FAQs. Thus UNIX novices might also want to browse the UNIX FAQs.

general UNIX questions

Sources for FAQs

news.answers

The best source of FAQ s of all kinds is a special newsgroup entitled `news.answers`. Many groups regularly post their FAQs here. In addition to FAQs on Linux, UNIX, and programming languages, a new explorer can find FAQs on other subjects of interest outside the realm of computer science.

directory doc

The special Linux FAQs are posted regularly in the newsgroup `comp.os.linux.announce`. Naturally, current FAQs can also be found on the many FTP servers, usually in a subdirectory `doc` under `Linux`. For example, the FTP server `nic.funet.fi` stores them in `/pub/OS/Linux/doc/FAQ`.

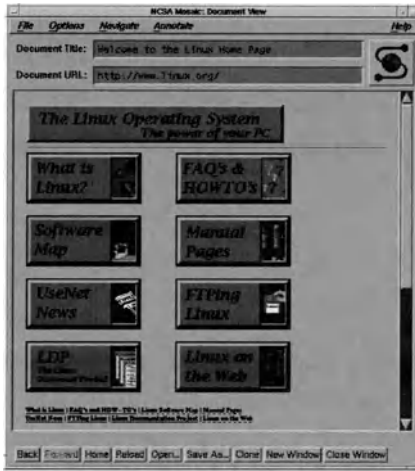


Figure 11.3. WWW site with references to Linux information

11.5 WWW

World Wide Web (WWW) is one of the most attractive and by now perhaps the most frequently used service on the Internet (see 10.11). There are many WWW server s for Linux as well, which offer FAQs, HOWTOs, manuals, and other information (see Figure 11.3).

WWW server

11.6 Mailing lists

Various mailing lists have been created for kernel hackers and other active Linux system collaborators. These mailing lists primarily support the exchange of news of developments, problems, ideas, and patches. Messages sent to the address of the mailing list are collected and forwarded several times a day to all addressees on the mailing list.

new developments

To join the mailing list for a certain subject, called a *channel*, the user sends a message containing the command `help` to the address `Majordomo@vger.rutgers.edu`; the reply will contain detailed instructions for using the mailing list. In the simplest case the following command suffices to request help for mailing lists:

channels

```
echo help | mail Majordomo@vger.rutgers.edu
```

Since these mailing lists are intended primarily for Linux developers, a novice should not pose questions through the normal channels.

11.7 Other documents

Manuals on Linux have been compiled by various authors from the Linux Documentation Project (LDP). Beyond the *Linux User's Guide* and the *Linux Installation Guide*, this documentation also includes more complex documents such as the *Network Administration Guide* (NAG), which covers almost all aspects of network installation under Linux. The *Kernel Hacker's Guide* (KHG) *Kernel Hacker's Guide* (KHG) deals with the structure of the kernel and the development of device drivers for Linux. It also provides an excellent glimpse into the inner workings of Linux.

These manuals can be procured via FTP and printed. Some book stores sell printed-out and bound copies of these manuals. The public domain files can be drawn from the FTP server `sunsite.unc.edu` in the directory `/pub/Linux/docs/LDP`.

11.8 Other sources

Besides the sources identified here, there are many other Internet services and organizations that provide information and document. This section identifies some of them.

WAIS

WAIS (Wide Area Information System) is an Internet service that supports searching WAIS servers worldwide for documents of all kinds by keywords. Through WAIS it is possible to obtain manuals and FAQs. Clients, a list of WAIS servers, and a detailed description of WAIS can be drawn per FTP from the host `think.com` or from the `comp.infosystems.wais` news group.

README files

As with most larger programs, the Linux kernel documentation is complemented by release notes and README files containing important instructions for its installation and operation. These files normally reside in the same directory as the system or the source code of the system.

release notes

source code

For example, the directory `/usr/src` contains many subdirectories with system components and utilities. Almost all of these subdirectories contain their own README files. Hence in the directory `/etc/lilo` we usually find the Linux Loader (LILO) along with a README file that describes both installation and configuration in detail.

Applications

An operating system without applications is only interesting for a very few. Linux has not been a system of that kind for a long time now. Aside from the vast number of free applications available from the Internet, there are also numerous commercial applications available. In addition, many applications for other PC UNIX versions can be run under Linux with the iBCS2 emulation. We can only present a small selection of all the available applications here.

Internet
applications

iBCS2

12.1 Desktop environment

Today's computer systems usually offer the user a graphical environment for the administration of files and programs. The file managers that are available for Linux at no cost are usually inferior to the commercial versions. The *Freedom Desktop*, which can be run on all the common UNIX platforms, has also been ported to Linux (see Figure 12.1). It should satisfy even sophisticated requirements. The program manager permits grouping individual files, programs, or shell scripts for a better overview. Files can be conveniently managed with the file manager. Both program manager and file manager support drag and drop. The Freedom package also includes several other additional utilities, such as a graphical front end for the `find` command and a print manager.

Freedom Desktop

drag & drop

12.2 Editors

Text- editing programs are one of the most important components of a computer system. Nearly a dozen such editors are available under

Texts

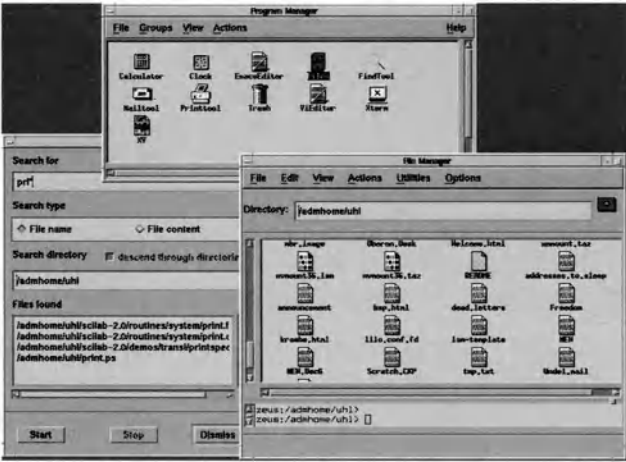


Figure 12.1. Freedom Desktop for Linux

Linux. Several of them are available on every UNIX system, while others have to be purchased separately with proprietary systems.

vi

The standard editor of every UNIX system is certainly *vi*. Since this is a rather aged program, it is no wonder that its user comfort level leaves room for improvement in several points. *vi*'s distinction between command and input mode irritates most users. Without a doubt, this approach has its advantages, but it also requires some adjustment. For example, it is possible to repeat the last command string and to replace the next three words in a step with a new word.

command mode

Since the source code of the *vi* editor is not available, a *vi* clone tends to be used under Linux. Even here there are two alternatives, *elvis* and *vim*, both of which emulate (almost) all commands of the original and provide several extensions as well.

elvis, vim

sed

Not an editor in the usual sense, *sed* is rather used as a stream editor to modify a data stream by means of commands in a control file. *sed*'s commands are largely compatible with those of *vi*, but the user does not input them interactively; instead, they are read from a file or from

stream editor

vi commands

the command line. `sed` often comes to play in UNIX shell script `s` or for processing extremely large files that normal editors cannot load.

shell scripts

joe

`joe` is short for “Joe’s own Editor.” This editor finds appreciation among converts from DOS because its extended keys rely strongly on that of the well-known PC word processing program Wordstar and the Turbo Pascal editor. The command and editing modes are not separate. Marked text blocks are displayed inversely, which is not always a given under UNIX. `joe` can divide the screen into multiple windows, format paragraphs, and use hyphenation mode.

changeover
Wordstar
blocks

People looking for a more powerful editor, who nevertheless prefer not to give up the Wordstar-compatible extended keys, should take a closer look at the Wordstar mode of the GNU Emacs editor.

Emacs

xedit

`xedit` is part of the X Window System package. Contrary to the above editors, `xedit` is not confined to an ASCII environment. Although `xedit` runs under a graphical user interface, it does not offer the level of comfort that would be expected; its available commands are essentially limited to the possibilities given by the Athena text widgets. While this editor is not likely ever to enjoy broad propagation, it does suffice for simple tasks.

X11

less comfort
Athena widgets

axe

`axe` is another editor that runs only under the X Window System. It also uses the Athena text widgets but provides much more functionality than `xedit`. The user can open an arbitrary number of text files in different windows, open and save files interactively via a file selection dialog, seek or replace text, and format paragraphs. Simple on-line help gives quick information on available commands. `axe` proves to be a multifaceted and user-friendly editor under Linux for the X Window System.

Athena widgets

several windows
seek / replace

xcoral

`xcoral` also requires the X Window System environment (see Figure 12.2). Although `xcoral` uses no standard toolkit, it has a pleasing



Figure 12.2. xcoral

menu bar
C++
browser

look with a menu bar and a scroll bar. `xcoral` particularly interests C and C++ programmers because of its integrated function and class browser s. When `xcoral` is started, it scans all C/C++ files and displays all the identifiers contained therein in an alphabetical list. From the browser the user can then branch to the respective location in the source code. In particular, this facilitates a programmer's familiarization with already-existing source code. For writing new programs, the programmer can select an optional mode that handles uniform formatting and can create function and class templates. Keyboard commands in `xcoral` bear a strong resemblance to the Emacs editor.

asedit

Motif
few functions

`asedit` uses the Motif widget set, which is reflected particularly in its consistent interface. Although this editor offers the most important commands, such as loading and saving texts and search-and-replace, it is quite Spartan otherwise. Its hypertext help deserves special mention.

Emacs and variants

ASCII and X11

The most popular editor for UNIX is certainly Emacs in all its variants. A major reason for its popularity is that it runs both on pure ASCII terminals and in a graphical X11 environment.

`lemacs` is a popular Emacs variant, developed by the Lucid company. It is based on an early version of Emacs 19. `lemacs` is distinguished primarily by its optical appearance and the syntax of special Lisp functions.

`lemacs`

With the release of Emacs version 19, other Emacs versions such as `xemacs` and `epoch` declined in importance. Hence we refrain from further detail on these. We will describe GNU Emacs in more detail in the next chapter.

GNU Emacs

12.3 Graphic programs

Many graphic programs are available for the X Window System. Their function extends from vector-oriented drawing programs to picture processing, and conversion. Only a few of these programs exist in a special Linux version. Usually they can be directly translated under Linux. Linux distributions already contain the most important of these programs.

drawing, painting
processing
and converting

xv

J. Bradley’s `xv`, a very powerful program for the display and manipulation of graphics of all kinds, enjoys wide dissemination in the UNIX world. It is available from the usual FTP servers, however, it is not free, but rather shareware. It has a very mature interface and is easy to use despite its many powerful features. `xv` loads and saves all common graphic formats such as GIF, TIFF, PostScript, JPEG, and PBM (see Figure 12.3).

shareware

graphic formats

A graphic to be loaded is selected from a file selection menu or (since version 3.00) via an integrated file manager that can display selected files as mini-pictures. The size of a loaded graphic can be changed freely. A graphic-processing menu permits the modification of numerous parameters such as brightness, contrast, and tone by means of numerous buttons and slide switches as well as mouse-modifiable curves.

file manager
scale

colors

A graphic loaded into `xv` can be modified in various ways, such as by using an algorithm like “Oil Painting” or “Emboss.” Furthermore, it can be displayed in various forms as a background graphic and is retained even after quitting the program.

modifications

background

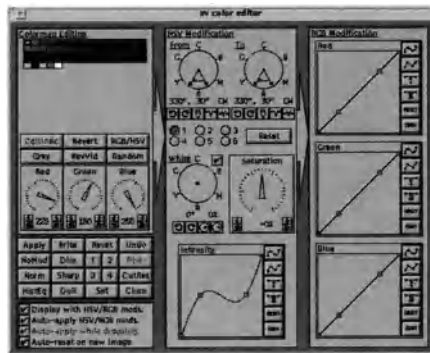


Figure 12.3. xv (Must find actual illo)

xfig

vector graphics

xfig, a program for creating vector graphics under Linux, offers all the usual drawing tools as well as the possibility of embedding text in various fonts (see Figure 12.5). One of its interesting features is

formats

the export of graphics in various formats. Besides its own fig format,

PostScript, HPGL, \TeX

xfig supports standards like PostScript, HPGL, and \TeX in variants.

idraw

InterViews

Stanford University's InterViews package contains a quite powerful, vector-oriented drawing program named **idraw** (see Figure 12.6). Besides the usual graphical elements such as lines, rectangles, circles,

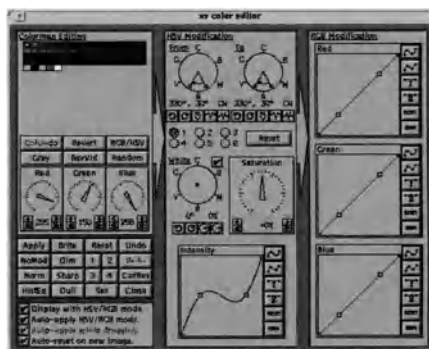


Figure 12.4. xv color editor

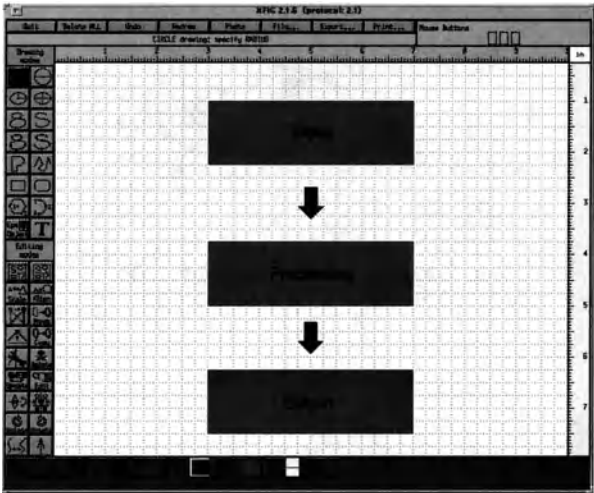


Figure 12.5. drawing program xfig

and Bézier curves, it also supports the free rotation of texts in various fonts.

Bézier curves

The only file format that this program supports is PostScript. Unfortunately, the program places high demands on the hardware, which makes the program uninteresting for 386 computers.

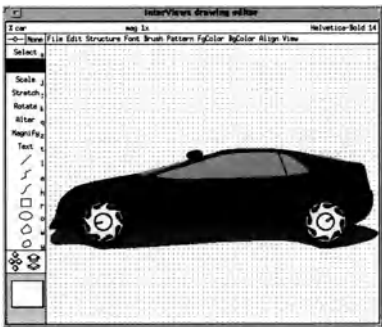


Figure 12.6. idraw



Figure 12.7. xpaint

xpaint

MacDraw xpaint is a program for the creation and manipulation of graphics that borrows from the Apple Macintosh MacDraw program. xpaint can process multiple graphics in different windows. Various tools support the drawing of any kind of figures: not only simple elements such as lines and circles, but also freehand curves as well (see Figure 12.7).

A fill-pattern editor permits the creation of additional patterns. The adjustable zoom function proves quite practical; it displays an excerpt from the graphic in a separate window. It is noteworthy that all drawing functions are supported in the zoom window as well.

fonts The xpaint font browser offers a clear selection of fonts, including scalable fonts as available under X11 R5 and R6.

Ghostscript/Ghostview

PostScript An important component of the Linux system is the PostScript interpreter Ghostscript (gs). Together with its graphical front end Ghostscript Ghostview, it permits the comfortable display of PostScript files (see Figure 12.8). But Ghostscript implements more than just the screen output; it also transforms an ordinary dot matrix or ink jet printer into a full PostScript printer.

dithering When color graphics are printed on a monochrome device, Ghostscript uses dithering to achieve reasonable quality. Ghostscript in the current version even supports PostScript Level 2.

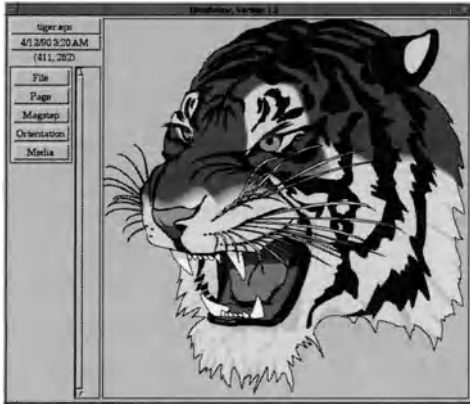


Figure 12.8. Ghostview with PS graphic

To be able to handle text that uses any of the copyrighted Adobe fonts, Ghostscript includes a package of similar fonts, but their quality is noticeably poor. Fortunately, Ghostscript also allows the installation and use of the original fonts.

Adobe

MPEG video player

Any of the many MPEG videos from the Internet can be viewed under Linux using the MPEG Player from the University of California at Berkeley (see Figure 12.9). Note that this requires no special hardware and that the output can be redirected to other screens. Unfortunately, the performance of the computer and its graphic board strongly influences the speed of the player.

no special hardware

Some videos from the Internet come in the Apple Quicktime format. The program `xanim` is needed to play these.



Figure 12.9. MPEG video player

12.4 Word processing

Most text-processing systems under Linux are not word processors. Generally, control sequences are inserted appropriately into ASCII text in order to influence the formatting of the text. This approach might seem quite inconvenient at first, but it does have its advantages, particularly for large documents.

groff

ASCII text The classical way to create formatted documents under a UNIX system is to process an ASCII text with the command `nroff`. This is a package for formatting texts, tables, formulas, and simple graphics. The user enters text in ASCII format and influences the layout with corresponding format commands. Output can be on a text-oriented or a graphical device.

Manual pages All UNIX On-Line Manual pages were created in this way. Normally the primary use of this utility is to display Manual pages. `nroff`'s functionality can be extended with external macros, and macro packages are available for various tasks. There is a dedicated macro file (`an`) for formatting a Manual page. The command looks like this:

```
zeus:/home/uhl> nroff -man /usr/man/man1/ls.1 | more
```

`nroff` Linux provides an enhancement rather than the original `nroff`. The command `nroff` is only a script that invokes `groff` with the correct parameters for ASCII output. If `groff` is invoked directly without these parameters, then PostScript text is output. This proves practical for printing Manual pages on a PostScript-compatible printer, since the text is properly formatted and output with various fonts.

```
linux2:/> groff -man /usr/local/man/scotty.1 >scotty.ps
```

T_EX

D. E. Knuth The typesetting system T_EX (pronounced “teck”) by Donald E. Knuth represents a world of its own. On first contact, the user of a modern

WYSIWYG word processing system may feel projected back to the stone age of computers. Nevertheless, \TeX affords some features that make it superior to normal word processing in several ways. One advantage of the system is its free availability for almost all computer platforms and thus the portability of the generated texts.

free availability

\TeX processes files that contain special formatting commands in ASCII format. The powerful program proves especially suitable for typesetting mathematical material, but the quality of normal texts also exceeds that of word processing programs. \TeX translates an ASCII input file to a DVI (device-independent) file, which can be displayed on the screen or printed. The format of the DVI file is identical on all computer systems, which makes transferring a \TeX file problem-free. The file can also be output to a linotronic machine.

DVI format

Numerous macro packages and auxiliary programs exist for \TeX that are also available under Linux. These include a graphical previewer (`xdvi`), a utility for sorting an index file, and a program to automatically generate missing fonts. There are also many drivers that convert DVI files to PostScript (`dvips`) or output them on non-PostScript printers (`dot matrix`, `ink jet`, `laser`). Graphics can be embedded with \LaTeX commands or with externally created PostScript files.

Previewer

One of the best-known macro packages is \LaTeX , by Leslie Lamport, which appreciably eases working with \TeX . Naturally, there are also versions for non-English texts. \LaTeX supports the automatic generation of a table of contents and an index. \LaTeX also eases the formatting of tables and lists. The following example shows a \LaTeX input file followed by its output:

 \LaTeX

```
\documentstyle{article}
\topmargin -15mm \headsep 0mm \textwidth 16cm
\textheight 26cm \oddsidemargin 0cm \parindent 0mm

\begin{document}
\thispagestyle{empty}

\centerline{{\Huge Typesetting with \TeX}}
\vspace{1cm}

\TeX\footnote{pronounced "teck'," or, better, with a Greek chi} and
the macro package \LaTeX\ enable the production of manuscripts in
typeset quality. It proves especially suitable for articles, books,
letters, mathematical material, and documentation. Particularly
\LaTeX\ provides numerous features for formatting formulas, tables
and lists. Tables of contents and scientific numbering of chapters can
be generated automatically. Even the management of footnotes proves to
be no problem. Various fonts and styles are available for emphasizing
text:
```

```

\begin{center}
{\rm Roman}, {\bf Bold Face}, {\tt Typewriter}, {\it Italic},
{\sl Slanted}, {\sc Small Caps}, {\sf Sans Serif}
\end{center}

The size of the text can also be varied:

\begin{center}
{\tiny tiny}, {\scriptsize very small}, {\footnotesize smaller},
{\small small}, {\normalsize normal}, {\large large}, {\Large larger}\
{\LARGE even larger}, {\huge huge}, {\Huge gigantic}
\end{center}

Mathematical formulas could look like this:
\begin{displaymath}
\int_0^\infty f(x)dx \approx \sum_{i=1}^n w_i e^{x_i} g(x_i)
\end{displaymath}

\begin{displaymath}
\sqrt[n]{\frac{x^n \cdot y^n}{1 + u^{2n}}}
\end{displaymath}

Here is an example of a list:

\begin{itemize}
\item Hardware
\begin{itemize}
\item Computer \item Keyboard \item Monitor
\end{itemize}
\item Software
\begin{itemize}
\item Operating system \item User interface \item Application program
\end{itemize}
\end{itemize}

Tables are especially easy to typeset under \LaTeX/:

\begin{center} \begin{tabular}{|r|l||c|rrr|c|c|} \hline
Rank & Team & & Sp. & S & U & N & Goals & Points\\ \hline \hline
1. & Bavaria Munich & 33 & 19 & 13 & 1 & & 66:31 & 51:15\\ \hline
2. & Hamburg & 33 & 18 & 9 & 6 & & 65:37 & 45:21\\ \hline
3. & Bor. M'Gladbach & 33 & 17 & 7 & 9 & & 70:44 & 41:25\\ \hline
4. & Bor. Dortmund & 33 & 14 & 10 & 9 & & 66:50 & 38:28\\ \hline
5. & Werder Bremen & 33 & 16 & 6 & 11 & & 63:53 & 38:28\\ \hline
6. & Kaiserslautern & 33 & 15 & 7 & 11 & & 64:47 & 37:29\\ \hline
\end{tabular} \end{center}

\newcommand{\absatz}{
\begin{minipage}[b]{7.5cm}
A text can also be wrapped in a text box. By the way, \TeX/ naturally
features automatic hyphenation. Furthermore, individual letters are
moved to certain positions under one another. This process is called
{\em kerning}.
\end{minipage}

\absatz \hfill \absatz

\begin{center}
{\Huge W/orld V/at V/A W/orld}
\end{center}
\begin{center}
{\Huge World Vat VA World}
\end{center}

\end{document}

```

After compilation, the result can be displayed with the command `xdvi`, with the result shown in Figure 12.10.

12.5 Multimedia environment Andrew

Developers at Carnegie-Mellon University have been working together with IBM for several years on the formation of an extensive,

T_EX¹ and the macro package L^AT_EX enable the production of manuscripts with typeset quality. It proves especially suitable for articles, books, letters, mathematical material, and documentation. Particularly L^AT_EX provides numerous features for formatting formulas, tables and lists. Tables of contents and scientific numbering of chapters can be generated automatically. Even the management of footnotes proves to be no problem.

Various fonts and styles are available for emphasizing text:

Roman, **Bold Face**, **Typewriter**, *Italic*, *Slanted*, SMALL CAPS, Sans Serif

The size of the text can also be varied:

tiny, very small, smaller, small, normal, large, larger
even larger, huge, gigantic

Mathematical formulas could look like this:

$$\int_0^\infty f(x) dx \approx \sum_{i=1}^n w_i e^{x_i} g(x_i)$$

$$\sqrt[n]{\frac{x^n - y^n}{1 + u^{2n}}}$$

Here is an example of a list:

- Hardware
 - Computer
 - Keyboard
 - Monitor
- Software
 - Operating system
 - User interface
 - Application program

Tables are especially easy to typeset under L^AT_EX:

Rank	Team	Sp.	S	U	N	Goals	Points
1.	Bavaria Munich	33	19	13	1	66:31	51:15
2.	Hamburg	33	18	9	6	65:37	45:21
3.	Bor. M'Gladbach	33	17	7	9	70:44	41:25
4.	Bor. Dortmund	33	14	10	9	66:50	38:28
5.	Werder Bremen	33	16	6	11	63:53	38:28
6.	Kaiserslautern	33	15	7	11	64:47	37:29

Text can also be wrapped in a text box. By the way, T_EX naturally features automatic hyphenation. Furthermore, individual letters are moved to certain positions near one another. This process is called *Kerning*.

Text can also be wrapped in a text box. By the way, T_EX naturally features automatic hyphenation. Furthermore, individual letters are moved to certain positions near one another. This process is called *Kerning*.

World Vat VA World
World Vat VA World

¹pronounced 'teck'

Figure 12.10. example of T_EX output

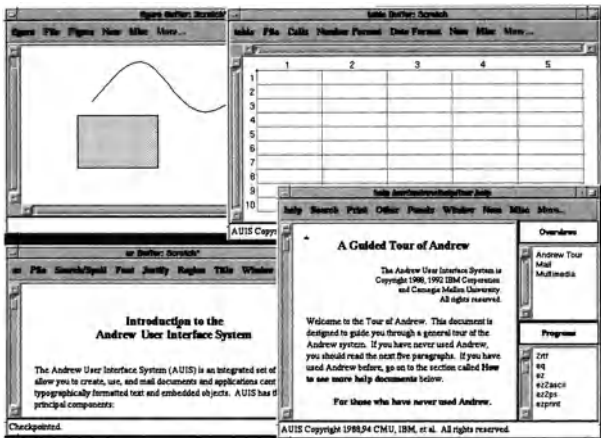


Figure 12.11. Andrew package

AUIS distributed multimedia user environment called AUIS (Andrew User Interface System). This refers to a series of individual applications that enable the processing of multimedia document (see Figure 12.11).

ez, table figure, image The applications include a simple text processor (ez), a spreadsheet (table), a drawing program (figure), and a painting program (image). A hypertext on-line help (help Andrew) documents the individual components.

mail A mail program capable of multimedia makes it possible to send the created documents. It is also possible to combine the individual components (text, graphics, tables, animations) in a document as needed. This process is often referred to as object embedding. Under AUIS the individual objects are called *insets*.

12.6 Databases

The following list of databases that run under Linux is only a small selection from a constantly growing group. Even systems for SCO UNIX, such as Foxpro for UNIX (Microsoft) and a commercial Ingres version, run under Linux with the help of the iBCS2 emulator. Of course, there is also a whole series of products that have been ported specifically for Linux. The Clipper-compatible database

Flagship may be interesting for anyone wishing to port software from the PC area to Linux.

MSQL

MSQL (mini SQL) was developed by David J. Hughes from the Bond University in Australia. It is a minimum database core for implementing a subset of the ANSI SQL standard. The available data types are currently restricted to character strings and whole or floating-point numbers. Unfortunately, MSQL does not yet recognize any Views. However, the system is probably sufficient for developing simple applications. Its performance is considerably better than that of other free database systems. In addition, the MSQL server can be addressed via a TCP/IP connection, which enables real client/server architectures. There is also a Tcl/Tk connection.

View

client/server

OBST

The object-oriented database OBST was developed within the framework of the STONE project (A Structured and Open Environment) at the Forschungszentrum Informatik (FZI) in Karlsruhe, Germany. Although this system was originally designed to be language-independent, the database description language is strongly oriented to C++. The program development, however, can be done in C, C++, and also in Tcl.

STONE

In addition to the rudimentary administration tools, a simple graphical database scheme editor is also available to the user.

A software company in Karlsruhe is currently developing a commercial OBST variant.

commercial version

Ingres and Postgres

The database system Ingres and its successor Postgres from the University of California at Berkeley have been ported to Linux. However, do not confuse this version of Ingres with the commercial SQL database system. The Linux version is significantly older and has a somewhat different query language instead of SQL.

SQL

Postgres is an object-oriented database, but unlike many of the proprietary systems, it is not based on C++ and persistent objects. It is rather an enhancement of the classical, relational approach.

object oriented

Both systems are quite interesting for educational purposes, but their usefulness is limited for developing applications.

YARD

- SQL Anyone needing a complete SQL database under Linux should look at the products from Yard Software in Cologne, Germany. YARD-SQL is a robust database system that corresponds to ANSI standard. Aside from an ESQL-C port, it provides secure transactions, referential integrity, and the data type BLOB (Binary Large Object).
- ESQL
- The graphical scheme editor can be used for database design. An
- ODBC ODBC port enables the connection of PC clients under MS-Windows.

Just Logic SQL

- low-cost SQL The Just Logic Technologies company has developed a client/server SQL database in the low-cost range. The server can run on all the common PC UNIX platforms. There are clients for DOS, Windows, and OS/2. An ODBC port and an Apple client are under development.

POET

- Another commercial database system for Linux is POET. This database system is C++-based and object-oriented; it has been ported to all the common operating system platforms (UNIX, OS/2, MS-Windows) and is useful for implementing client/server solutions. Since POET is directed exclusively at C++ programmers, it only has a very rudimentary user interface.
- C++
- client/server

12.7 Mathematical applications

- MuPAD A particularly interesting program package for students is MuPAD (see Figure 12.12). This computer algebra system was developed at the University of Paderborn (Germany) and is furnished free of charge to non-commercial institutions. MuPAD supports the user in solving various types of mathematical problems. An integrated programming language enables the implementation of the user's own algorithms. MuPAD was especially designed to handle parallel problems.
- algebra

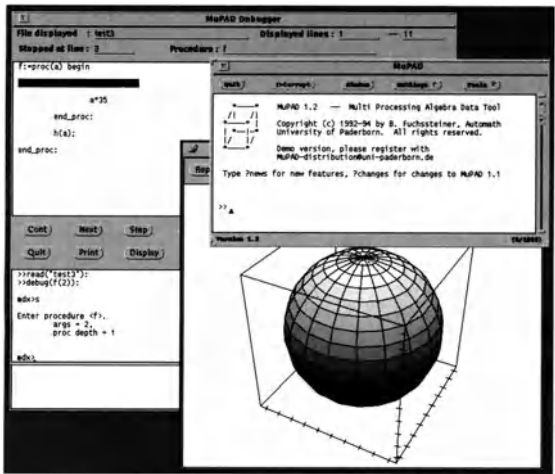


Figure 12.12. Computer algebra system MuPAD

MuPAD provides its own debugger with a graphical front end. MuPAD itself has a simple command-line-oriented user interface. However, an OpenLook- or Motif-based front end makes the program appreciably easier to use. MuPAD's graphical output features, e.g., for depicting three-dimensional functions, are quite advanced.

debugger
OpenLook/Motif
on-line help

A hypertext on-line help function makes it possible to access all relevant information and function descriptions. A detailed manual for MuPAD was published by Birkhäuser.

Maple V

Waterloo Software has ported Version 3 of the well-known Maple V package to Linux (see Figure 12.13). This system may well satisfy some users' exacting demands, but it is expensive.

Waterloo Software

12.8 Simulations

The Simulator for Neural Networks (SNNS) developed by computer engineers at the University of Stuttgart (Germany) is one of the best for this sector (see Figure 12.14). The graphical front end makes the development and analysis of more complex networks considerably easier. SNNS is extremely useful not only for demonstrating the

SNNS

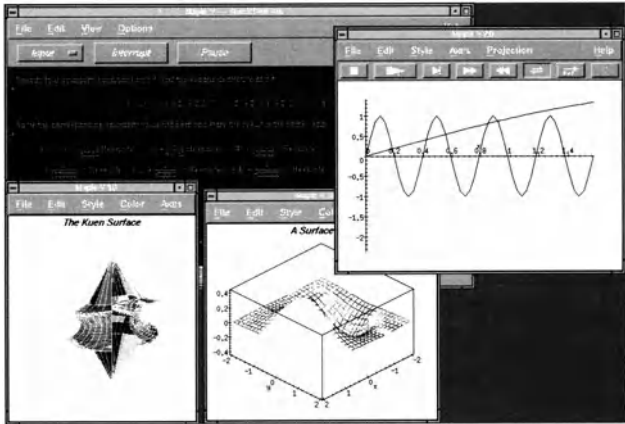


Figure 12.13. Maple V under Linux

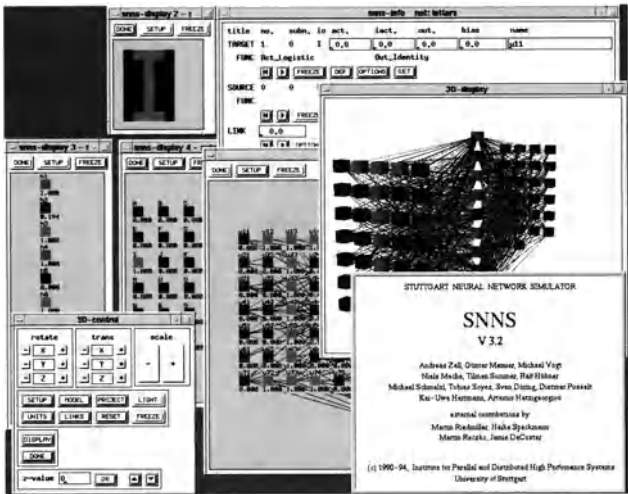


Figure 12.14. Simulation of neural networks with SNNS

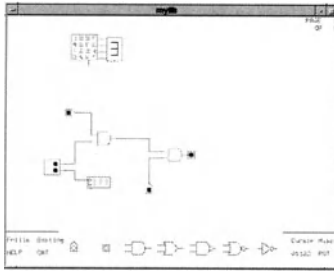


Figure 12.15. digital circuitry simulation

processes in neural networks, but also for training networks that are actually usable.

Circuitry simulation

The Caltech Electronic CAD Distribution is useful for simulating digital and electronic circuitry (see Figure 12.15). Digital circuits can be built up and tested with `diglog`. In addition, a comprehensive library of components and circuit elements is available.

12.9 Games

To relax after a day's work, the user can enjoy one of the many games available under Linux.

XTeddy

XTeddy is not a game in the real sense, but it is nevertheless an indispensable program (see Figure 12.16). It is particularly popular among family members who are not (yet) computer-dependent.



Figure 12.16. electronic teddy bear

playing pieces

Tetris

Tetris is an extremely popular game that is also available under Linux in an attractive form. The player must try to place falling blocks in such a way that they do not pile up too high. This would happen quickly, of course, if completed rows did not disappear.

GNU chess

For more challenging entertainment, have a look at `xboard` and GNU Chess. The chess skill of this program is astonishing. At chess competitions, GNU Chess has defeated commercial chess programs.

GNU Chess itself is not particularly user-friendly, so it requires the graphical front end `xboard` to make playing practical. `xboard` can also start GNU Chess twice and let the games play against each other. Another option allows chess parties on the Internet, making it possible to find an adequate partner somewhere in the world at any time, day or night (see section 10.3).

MUDs and Crossfire

Games that can be played on a network by multiple participants prove especially interesting. One widespread form of such network games is MUDs (Multi-User Dungeons) (see Figure 12.17). Each player logs in to a common server via telnet and can use commands like `say`, `get`, or `go`, and can move. MUDs are normally limited to textual input and output.

Some addresses of MUDs that can be reached by telnet are (see section 10.3):

- `morgen.cs.tu-berlin.de`, Port 7680
- `padermud.uni-paderborn.de`, Port 3000
- `pascal.uni-muenster.de`, Port 4711
- `infosgi.rus.uni-stuttgart.de`, Port 3333
- `mud.uni-munester.de`, Port 4711

newsgroups

Further information is available in the many newsgroups on MUDs and in the MUD FAQ (see section 11.4).

Crossfire

A leap ahead of the text-oriented MUDs, the multi-user role-play Crossfire requires special client software that graphically depicts

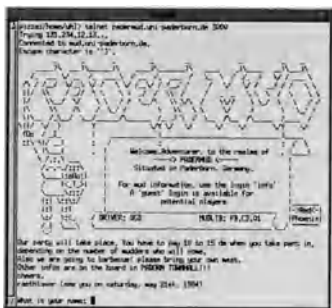


Figure 12.17. MUDs

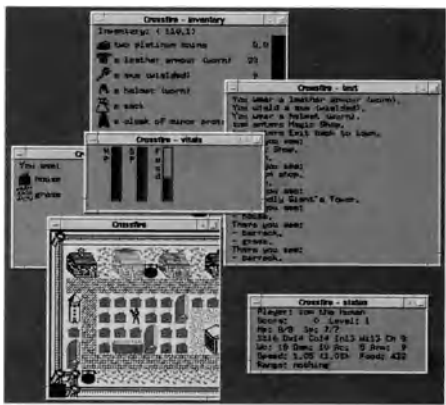


Figure 12.18. Crossfire

the virtual world in which the player moves (see Figure 12.18). The source code of the game can be copied from the FTP server `ftp.ifi.uio.no` in the directory `/pub/crossfire`.

GNU Emacs

If you ask several experienced UNIX users which editor they use, most of them will answer either vi or Emacs. Most vi users will base their choice on having used this editor for 20 years and will point out that it is available on all UNIX platforms. The Emacs users, on the other hand, will point out that their editor is extremely flexible and functional.

vi or Emacs

This chapter introduces the application of Emacs and describes its concepts and features in more detail. A section on programming with Emacs Lisp explains the configuration by means of examples. The last section may be difficult to understand for readers who do not already have some knowledge of other programming languages. A more in-depth presentation, however, would by far exceed the scope of this book. The last part of this chapter therefore addresses readers who already have a basic knowledge of programming.

application

configuration

13.1 Overview

Richard Stallman, the founder of FSF, developed Emacs, which is available on nearly all UNIX platforms, DOS, and VMS. Emacs can be used not only in text mode on simple ASCII terminals, but also under the X Window System. This system provides pulldown menus, buttons, and scrollbars, and various colors and fonts. With the mouse the user can select, copy, and insert text as usual under X11.

FSF

X11

Emacs provides a tutorial for novices as well as help for all the commands, key combinations, and internal variables. The complete documentation can be called up on-line in a form similar to hypertext.

tutorial

With Emacs the user can work on different texts simultaneously in separate or divided windows. A backup copy of the texts is

automatic backup copy

automatically stored at regular intervals. The almost unlimited undo function is particularly remarkable. With the touch of a key, the user can undo any number of changes up to a definable limit.

Unlike `vi` and many other well-known editors, Emacs is much more than just an editor. It contains an interpreter for its own Lisp-based language, which is also available to the user. Almost all the functions that go beyond primitive editing functions are written in this Emacs Lisp, or Elisp for short. There are Emacs Lisp programs, for instance, which can create a mail front end, a newsreader, or a complete developing environment from Emacs.

Due to its capabilities, the Emacs package is relatively extensive. The basic installation requires about 20 MB of memory on the hard disk. The basic installation contains not only many Emacs Lisp programs, but also the complete documentation.

With Emacs' numerous available major modes, the user can optimize the editor for specific tasks. The major modes support the programmer or author, for example, in formatting (source) texts. Directions in C and Lisp mode are automatically indented according to their box level. When the user enters a closing parenthesis, the appropriate opening parenthesis is automatically highlighted. Keywords, commentaries, or other language elements can be displayed in a different font or color if the user so wishes. These kinds of modes exist for all propagated programming languages and file types and are easy to customize.

13.2 Basic terms

Emacs uses some terms in the documentation and for names of functions, which other editors either do not have or use somewhat differently. We briefly explain those terms here in order to avoid confusion later.

The location between the user's current position in the text (the cursor) and the character before that is called the *point*. Aside from this, there is another important position, which the user can set explicitly; this is called the *mark*. The *region* is the area between the point and the mark. The region corresponds to a marked block in other editors. It can be deleted, cut out, or copied. Unlike in other



Figure 13.1. Emacs with multiple frames and windows

editors, in Emacs the region is always linked to the current cursor position (point).

A keyboard configuration is called *key binding* in Emacs, because it defines the connection between a key sequence and a specific function. Multiple bindings are stored together in a *keymap*. When a user works on a file in Emacs, work takes place in a *buffer*. A buffer usually contains the contents of a file, but it can also contain something completely different. Some examples of this are the input and output of the GNU debugger *gdb* when it is invoked under Emacs, or the contents of a directory if the user is in *direx* mode, explained ahead.

If a buffer is visible, it is displayed in a window. In Emacs jargon, however, a window does not correspond to a window under X11, but rather refers to a separate portion of the current window in which a buffer is displayed. A separate window under X11 is called a *frame* in this case, and it can contain several windows (see Figure 13.1).

13.3 Operation

As with any other editor, the user can invoke Emacs with a file name as parameter or without specifications. To work on the file `.cshrc` in the home directory, for example, invoke Emacs as follows:

key binding

keymap

buffer

window

frame

invocation

invocation

```
emacs ~/.cshrc
```

X11 Under X11, Emacs opens a new window on startup with a graphical menu bar. If Emacs is not invoked under X11, but rather in text mode, there is no menu bar and Emacs uses the entire screen of the terminal.

The user can operate Emacs with the cursor keys and function keys and also with special **<strg>** or **<meta>** key combinations. We explain the configuration of the function keys in section 13.8.

meta There is a **<meta>** key on the keyboards of some workstations, but not on the MF2 keyboards common to PCs and not on simple ASCII terminals. The **<esc>** key can be used instead of **<meta>**. For **<meta-x>**, press the **<esc>** key, then press **<x>** on keyboards without **<meta>**. The **<alt>** key is often redefined as **<meta>** under X11 (see section 8.8, which makes the **<esc>** route unnecessary.

The most important bindings of the standard keymap (without the function keys) are as follows:

Cursor movements:

Strg-f	character forward
Strg-b	character backward
Strg-n	next line
Strg-p	previous line
Strg-a	beginning of line
Strg-e	end of line
Strg-v	scroll page forward
Meta-v	scroll page backward

Delete:

del	Delete character before the cursor (backspace).
Strg-d	Delete character under the cursor.
Strg-k	Delete to end of line or blank line. This stores the deleted text in the <i>kill ring</i> , and it can be inserted again.

Mark, delete, and insert:

Strg-Space	Set the mark at the current position of the point.
Meta-w	Copy the region (text between marking and point) to the kill ring
Strg-w	Copy the contents of the region to the kill ring and delete it from the text.
Strg-y	Insert the contents of the kill ring at the current cursor position.

Search:

Strg-s	search (by increments, see below)
Strg-r	search backwards (by increments)
Strg-g	interrupt search or the last command

Miscellaneous:

Strg-x f	load file (find)
Strg-x Strg-c	end Emacs
Strg-x o	go to next window (other)
Strg-x b	go to a different buffer

<strg-g> is a relatively important key. A beginner often feels somewhat lost in Emacs' combinations of multiple keys. If the user accidentally presses **<strg-x>**, for example, Emacs expects a second entry, which in conjunction with **<strg-x>** could result in a command. This can be aborted by pressing **<strg-g>**.

interrupt

The user can find a clear and relatively complete list of all the Emacs key bindings in the Emacs reference card, which is stored in the Emacs `etc` directory. If Emacs is installed under `/usr/lib/emacs/19.28`, for example, then the `etc` directory is `/usr/lib/emacs/19.28/etc`. The reference card is named `refcard.ps` or `refcard.tex` as a \TeX file.

reference card

In addition, in Emacs the user can output a list of all key configurations at any time with **<strg-h>b** ("b" for binding) or the

keyboard configuration

meaning of a special key combination with **<strg-h>k<relevant key combination>** (“k” for key).

search The incremental search within Emacs is a feature unknown to many other editors. It means that Emacs does not wait until the user has the entire text for the search; as soon as the first key is pressed, Emacs begins searching for the next occurrence of that character and positions the cursor at the appropriate place. If the user enters another character, Emacs looks for the next occurrence of these two characters together. This way, the user approaches the relevant text incrementally and may find it with far fewer keys than by having to enter the entire search term.

menus Pulldown menus are available under X11 for the most important functions. In addition, all the (interactive) functions intended for the user can be invoked directly, regardless of whether they have been assigned to a certain key. Press **<meta-x>** for this. Emacs then prompts the user to enter the name of the function to be invoked with a prompt in the minibuffer. An automatic name extension is provided with **<tab>**, as is almost always the case in Emacs. If more information is required for the function to be invoked, a query is made in the minibuffer. It is often practical to invoke less frequently used functions directly with **<meta-x>**, especially if the user is not sure whether there is a binding for them at all.

13.4 Documentation and help

tutorial The Emacs tutorial provides a quick introduction to the operation of Emacs. The tutorial is a text that is automatically loaded when the user enters **<strg-h><t>**. It contains directions that interactively present all the important editing commands step by step. The actual functions on-line help consists of a number of partial functions. The user can display the description of a function with **<strg-h><f>**; **<strg-h a>** (a for *apropos*) searches a list of all the names of functions for a partial string (see Figure 13.2). For example, if the user is looking for a function in order to go directly to a particular line, then **<strg-h a>line** produces a list of all the functions containing *line* in their names, together with all the bindings that may be present. This list would contain such functions as *previous-line* and *next-line*, as well as the relevant function *goto-line*.



Figure 13.2. The apropos function

With **<strg-h b>** the user can display a list of all current key bindings, and **<strg-h m>** produces a description of the activated modes (see below). All of these help functions are also available in the help pulldown menu under X11.

Strg-h a	apropos – list of all the functions whose names contain a particular text
Strg-h c	brief description of a key combination
Strg-h k	longer description of a key combination
Strg-h Strg-k	go into the info mode to the page where the command is described that is linked to the key combination
Strg-h i	go into the info mode
Strg-h m	mode – description of the activated modes
Strg-h f	description of an Emacs Lisp function
Strg-h v	description of an Emacs Lisp variable
Strg-h h	brief overview of all strg-h commands

Table of the most important help functions

As with all FSF programs, the comprehensive documentation of the Emacs editor is originally in `texinfo` format. Info files are automatically created from this format on installation. Not only Info files, but also \TeX files can be created from a `texinfo` file. This means that the documentation can also be printed out (see section 12.4). Info files contain a hierarchic structure with cross references.



Figure 13.3. The Info mode in Emacs

Info mode The user can view them with Emacs in the Info mode (see Figure 13.3) or with another Info reader such as `tkinfo`.

cross references There is a new keymap in the Info mode that enables the user to follow cross references with simple keys or with the mouse and thus to navigate hierarchically in an Info file. Under X11 the cross references are emphasized by the use of a different font or color.

manual The information on Emacs that is displayed in the Info mode corresponds to the printed Emacs manual, which can be obtained from FSF. The most important keys in the Info mode are as follows:

middle mouse key	Follow the cross reference under the mouse key.
d	Go to the table of contents with the overview of all Info files (directory).
Enter	Follow the cross reference under the cursor.
l	Go back to the last page viewed (last).
n	Go to the next Info page (next).
p	Go to the previous Info page (previous).
u	Go to the Info page that is above the current page in the hierarchy (up).
s	Search for text (search).

In addition to the normal Emacs manual, there is also a documentation for Emacs Lisp, which is also available in `texinfo` format. This manual has many hundreds of pages when printed, and it comprehensively describes all the internal details of the editor and its programming. The user can obtain this manual via FTP, for example, from any larger FTP server. The file is normally stored together with the Emacs source code and the other GNU utilities in a directory named `gnu`.

Emacs Lisp

13.5 Modes

One major mode and possibly one or more minor modes are assigned to every buffer. A major mode generally defines a new keymap and often provides new functions that set Emacs for a specific special function. Thus there are major modes for almost all common programming languages, but also for editing configuration files and normal text. Some modes are not intended for editing files, but rather use Emacs for other functions. Some examples of this are the `gnus`, `rmail`, and `dired` modes.

major mode and
minor mode

A minor mode provides certain features that can be activated in addition to the selected major mode. One example is the font-lock mode, which highlights commentaries and keywords of programming languages with different fonts and colors.

The table below lists examples of some major and minor modes along with a brief description of their most important functions.

modes

Major modes:

c, c++	automatic indentation of the source text
lisp	automatic indentation of the source text, on-line help, evaluation of expressions
tcl	automatic indentation of the source text, evaluation of regions under Tcl

Minor modes:

font-lock	presents commentaries, keywords, strings, or other text areas in different fonts or colors
outline	inserts or removes text on different hierarchical levels
auto-fill	automatic word separation

loading files are loaded. The name of the major mode is displayed in the mode line of the corresponding window. The recognition characteristics for file extensions the mode selection are file names or file endings such as `.c`, `.cc`, or `.tcl`. These are assigned to their corresponding mode in a table. In addition, Emacs can determine the mode by the contents of the first line. In this case, it recognizes special commentaries such as `#!/usr/bin/wish`, `# -*-Tcl-*-`, or `# -*-Mode: Tcl;-*-`. The user can also activate a mode manually. To do so, press **<meta-X>** followed by the name of the mode, for example, **<meta-x>tcl-mode**.

13.6 Packages and enhancements

Lisp Aside from the modes, there are also Emacs Lisp programs that provide general additional functions. They can also use the Lisp interpreter only to execute relatively independent applications. We will explain some of these enhancements in the following subsections.

position of the cursor **saveplace** `saveplace` saves the position of the cursor when a working file is closed. The next time this file is loaded, Emacs automatically places the cursor in the same position it was in when the file was closed. The user can activate this function generally for all files or only for certain files. The corresponding entry in the Emacs configuration file is

```
(load "saveplace")
(define-key ctl-x-map "p" 'toggle-save-place)
```



Figure 13.4. The dired mode in Emacs

dired

Although `dired` is actually a mode, it does not have much in common with a normal major mode for processing a file. As the name suggests, `dired` enables the user to work on directories. Files can be copied, renamed, or deleted (see Figure 13.4).

files and directories

The key **f** (follow) permits branching into a subdirectory or loading a file into a different buffer. **dired** is automatically invoked when a directory is “loaded” instead of a file. The user can also activate it directly with **<meta-x> dired**.

follow

ediff

`ediff` is a valuable aid in combining different versions of a text. This Emacs Lisp program opens two files and displays them either next to one another horizontally or vertically as selected. The differences between the two files are highlighted with different fonts or colors and can be accepted by either file from the other with a key entry (see Figure 13.5). `ediff` can be selected from the menu or invoked with **<meta-x>ediff**.

versions

differences

finder and lispsdir

It is not easy for users to find their way around among all the Lisp programs and Emacs modes. The Emacs Finder program

programs



Figure 13.5. ediff in Emacs

can be helpful here. It groups the Lisp programs of the Emacs
selection distribution under keywords and displays them in a selection
list. If the user selects a keyword with **<space>**, then all the
packages that correspond to the keyword are displayed with a
brief description (see Figure 13.6). To start the Finder, load it
load library first with **<meta-x>** load-library. Then start the program with
<meta-x> finder-by-keyword.

The Finder knows only the Lisp programs that the Emacs
package already contains. The user can use the GNU Emacs Lisp
Code Directory to get an overview of other Emacs programs. This
is a list with over 800 different Emacs Lisp packages, which can be
searched with a special program. The current version of this list can



Figure 13.6. The Finder



Figure 13.7. Result of a `lispdir-query` on "tex"

be obtained from the server `archive.cis.ohio-state.edu` from the directory `/pub/gnu/emacs/elisp-archive`. The program for searching the archive is in the same directory as `lispdir.el.z` (see Figure 13.7).

func-menu

The `func-menu` program is very practical for programming in C. It analyzes the C source code in the active buffer at a mouse click and displays a pop-up menu with all the defined function names (see Figure 13.8). If the user selects one of these function names from the pop-up menu, Emacs jumps to the buffer location where the function is defined.

Tags

Although the management of tags is less convenient, it is functionally superior. The command `etags` creates a file in which all the function names are listed along with the file and the line where the function is defined. The user can then use Emacs Lisp functions such as `tags-search` and `tags-apropos` to search this list.

etags

search

In fact, the most important fundamental structure of a Lisp program is a list delimited by parentheses. This means that expressions frequently consist of five or more nested levels of parentheses.

lists

However, do not let this be a deterrant. You quickly adjust to the look, and with the support of Emacs Lisp mode, which automatically indents expressions correctly and displays the pairing of opening and closing parentheses, there is never any temptation to count parentheses.

modes

Symbols

Symbols play a particular role in Lisp. A symbol represents a unique name that is used for access to data, functions, and property lists. (We do not handle property lists in detail here.) Imagine a symbol as a small container with several compartments: one each for the name, the value, the function definition, and the property list.

shelves

name
value
function
property list

The value of a symbol has a certain type, which need not be declared in advance as in many procedural languages (e.g., Pascal) but is implicitly contained in the value. Emacs Lisp knows the type of each value. This means that the use of a wrong type can be detected only at run time.

type

The most important primitive data types in Emacs Lisp are

primitive types

- Integer
- Floating point
- Character
- Array
- Strings (one-dimensional array)
- Symbol
- Function

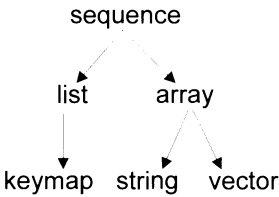


Figure 13.9. Excerpt from the type hierarchy in Emacs Lisp

- Macro
- Primitive function (written in C)
- Byte-code function (compiled function)

Beyond these data types, which also exist in other Lisp dialects, there are special data types in Emacs Lisp that are used for editor functions. The most important of these are

- buffer
- window
- frame
- process
- stream
- keymap

The data types in Emacs Lisp partially overlap; a value can simultaneously belong to multiple types. This is because some types are a specialization of other types. The data type `string`, for example, represents a specialization of the type `array`, which in turn is a specialization of the type `sequence`.

The values *true* and *false* as results of comparison or Boolean expressions are represented in Lisp with the special symbols `t` and `nil`. `nil` is a synonym for the empty list `()` and is thus also *atomic* (see below) and a list.

The interpreter

The Lisp interpreter executes a read-evaluate-print cycle. This means that it reads an input, tries to evaluate it, and then prints the

result. The expressions that the interpreter can evaluate are called *forms*, or *s-expressions*. The latter is a collective term for elements such as numbers, strings, and symbols that are designated as *atoms*, and *lists*.

If the interpreter encounters a value such as a string or a number, then the result of the evaluation is the value itself. If the interpreter encounters a symbol, it outputs the contents of the compartment with the value of the symbol. The following example shows several inputs and the respective output of the Lisp interpreter.

numbers

symbols

```
1
1
"test"
"test"
fill-column
78
```

If the interpreter encounters a list, then interpretation depends on the first symbol in the list. The symbol is not evaluated, but it determines the kind of s-expression. If it is the name of a function, then the list is a function call. The interpreter then attempts to evaluate the remaining elements (all except the first) and to invoke the function with the results of the evaluation (see Figure 13.10).

lists

function

In addition to lists that are interpreted as function calls, there are macros and special forms. In these cases the remaining list elements are not evaluated first. Special forms, for example, are conditions like `if` or `cond`.

macros and

special forms

A function call takes the following form:

function call

```
(functionName Argument Argument ... )
```

This is a prefix notation as we know it from mathematics. The following example shows several function calls and the output of the interpreter.

prefix notation

```
(+ 1 4)
5
(substring "abcdefg" 2 3)
"c"
(* (+ 2 3) 3)
15
```

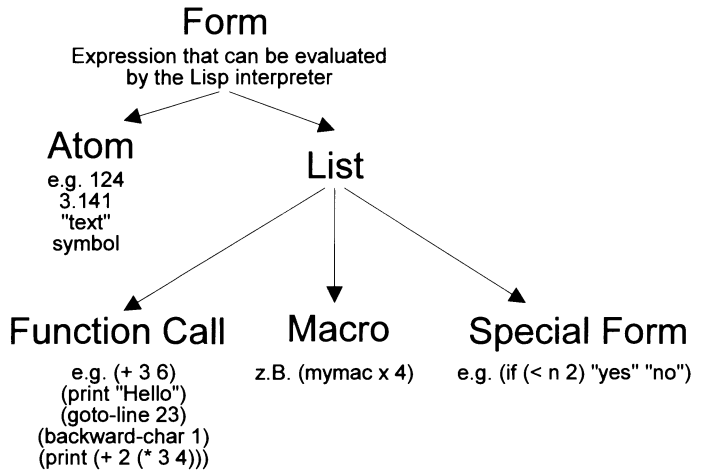


Figure 13.10. Expressions in Emacs Lisp

arguments

The individual arguments of the function calls can be symbols or lists, which again are interpreted as function calls. The evaluation of such a list is thus recursive.

special forms

Before we explain existing Lisp functions in more detail, we need to examine several of the special forms. Among other things, they help in realizing conditions and loops and in defining new functions.

Special forms

<code>(defun fname (P1...Pn) comment Form1 ... Formn)</code>	Defines function <code>fname</code> with parameters <code>P1</code> to <code>Pn</code> and function definition <code>Form1</code> to <code>Formn</code> .
<code>(defvar symbol value comment)</code>	Assigns to the symbol <code>symbol</code> a value if it has none; this is usually used in the definition of global variables.

<code>(if exp true false1 ... false2)</code>	Evaluates the form <code>exp</code> . If its result is not <code>nil</code> , the form evaluates to <code>true</code> and this result is returned. Otherwise the forms <code>false1</code> to <code>false2</code> are evaluated. The result is then the result of form <code>false2</code> .
<code>(cond (exp1 forms1) (exp2 forms2) ... (expn formsn))</code>	Similar to a case statement in Pascal, the expressions <code>exp1</code> to <code>expn</code> are sequentially evaluated until one is not <code>nil</code> . In this case the associated forms are evaluated; for <code>exp2</code> this would be the forms <code>forms2</code> . The result of the last associated form is returned as the result.
<code>(while cond forms)</code>	Loop: The forms <code>forms</code> are evaluated sequentially until the condition <code>cond</code> produces a value other than <code>nil</code> .
<code>(quote list) or 'list</code>	Returns <code>list</code> as a list without attempting to evaluate it.
<code>(progn forms)</code>	The forms <code>forms</code> are evaluated sequentially and the last result is returned as the result of the special form.
<code>(setq sname nvalue)</code>	Assigns the new value <code>nvalue</code> to the value compartment of symbol <code>sname</code> .
<code>(let (Var1 ... Varn) forms)</code>	Creates new links for variables <code>Var1</code> to <code>Varn</code> , which can then be used within the forms as local variables. <code>Var1</code> to <code>Varn</code>

	can be either symbols, in which case they receive the value <code>nil</code> , or lists of type <code>Var form</code> , in which case the symbol <code>Var</code> is linked to the result of <code>form</code> .
(<code>save-excursion forms</code>)	The forms <code>forms</code> are evaluated sequentially as with <code>progn</code> , but the current state of the editor, i.e., the current buffer, the cursor position, and the position of the marker are saved and restored after evaluation of <code>forms</code> .

Additional special forms are explained when they are used in an example. An overview of all special forms in Emacs Lisp is in the Info document for Emacs Lisp.

Examples of special forms

```
(defvar TestVariable 123
  "This variable is only for test purposes.")
(setq TestVariable (+ TestVariable 2))
```

- global variables Here the symbol `TestVariable` is used as a global variable. First the variable is assigned the value 123. Then the `setq` form increments the value by 2. The result of this form is the new value of `TestVariable`, or 125.
- testing It is quite simple to test these small examples in Emacs. Start Emacs without a file name. Emacs displays the usual version number and the help commands. This message disappears when any key is pressed, presenting a buffer named `*scratch*`. The current mode is “Lisp interaction,” which indicates that the Lisp interpreter is ready. The mode contains key bindings for working with Emacs Lisp:
- Lisp interaction

Tab	correct indentation of current line depending on nesting depth.
Meta-Tab	completion of the current symbol. Function names, variables and special forms are automatically extended.
Meta-Ctrl-x	evaluates the <code>defun</code> special form to the left of the cursor.
LFD (line feed key or Ctrl-j)	Evaluates the form to the left of the cursor and outputs the result.

Unfortunately, PCs lack a special linefeed key. However, the function can be assigned to another key with the invocation of **<Meta-x>** `local-set-key`. Emacs then prompts with "Set key locally: -" for the key combination to be assigned. Here, for example, we might press **<Meta-Return>**.

line feed
local-set-key

Emacs then asks for the function to be bound to the entered key combination. Here we specify `eval-print-last-sexp`. This assigns to the key combination **<Meta-Return>** the Emacs Lisp function `eval-print-last-sexp`, which evaluates the form to the left of the cursor and outputs the result. After entering a form, we can now simply enter **<Meta-Return>** instead of **<Return>**, and the Emacs Lisp interpreter evaluates the form. The result for the above example looks like this:

function
meta return

```
(defvar TestVariable 123                                <Return>
  "This variable is only for test purposes.")           <Meta-Return>
TestVariable
(setq TestVariable (+ TestVariable 2))                  <Meta-Return>
125
```

In the following example a simple function is defined that multiplies the sum of two numbers by 2. Then the new function is used in an invocation of `print`:

```
(defun doublesum (a b)
  "simple test function"
  (* 2 (+ a b)))
doublesum
```

```
(print (doublesum 2 3))
10
10
```

double output The Lisp interpreter outputs the result twice: once due to the function `print`, and again as the result of the overall form in which `print` was invoked.

doc string The preceding examples each specify a string as comment. This comment, also called a *doc string*, is saved along with the definition.

describe function These comments can later be read using the predefined functions `describe-function` and `describe-variable`, which are bound to `<Ctrl-h><Ctrl-f>` and `<Ctrl-h><Ctrl-v>`, respectively. This means that the same help function can be used for custom functions as was employed for predefined functions.

local variable During programing, some intermediate result is needed in several subsequent steps, and thus it is stored in a local variable. In Lisp this works with the special form `let`, as the following example shows.

```
(defvar square 0
  "global value of square")
square

(defun f (n)
  "example function using let"
  (let ((square (* n n)))
    (print square)
    (print (- square 2))))
f
(f 3)

9
7
7

square
0
```

`let` The `let` form creates a new binding for the symbol `square`. This covers the global value of the symbol. The new binding only applies locally within the `let` form. On leaving the `let` form, the global value of `square`, which has not changed, applies again.

dynamic binding Contrary to the late binding employed in Common Lisp, Emacs Lisp uses dynamic binding. This means that the values of new `let`

variables are accessible from within functions that are invoked within `let`. The following example clarifies this:

```
(defvar a 100)
a
(defvar b 200)
b
(defvar c 300)
c
a
100
(defun fct1 ()
  "Test function using let"
  (print "in fct1")
  (print (format "a=%s, b=%s, c=%s" a b c))
  (let ((a 1) (b (+ 2 3)) c)
    (print "in fct1 in let")
    (print (format "a=%s, b=%s, c=%s" a b c))
    (fct2))
  (print "in fct1 again outside of let")
  (print (format "a=%s, b=%s, c=%s" a b c)))
fct1
(defun fct2 ()
  "Test function, invoked by fct1 and accessing variables"
  from fct1"
  (print "in fct2")
  (print (format "a=%s, b=%s, c=%s" a b c)))
fct2
(fct1)

"in fct1"

"a=100, b=200, c=300"

"in fct1 in let"

"a=1, b=5, c=nil"

"in fct2"

"a=1, b=5, c=nil"

"in fct1 again outside of let"

"a=100, b=200, c=300"
"a=100, b=200, c=300"
```

Type-check functions

Predefined functions are a significant component of Emacs Lisp. Due to their sheer number, we can present only a small fraction of them. However, the help functions in Emacs and the Info files for Emacs Lisp provide a very practical reference of the available functions.

The fact that in Lisp the type of a symbol or a function return value can often be determined only at run time necessitates functions for type checking. The following table lists examples of such functions, which test whether the passed expression is of a certain type and then return `t` or `nil`. Their names usually derive from the name of the type with the extension `p` for predicate. Lisp

functions

type

`t` or `nil`

frequently uses this extension for functions that execute a test and return `t` or `nil`.

<code>arrayp</code>	tests whether the passed value is an array
<code>bufferp</code>	tests whether the value for a buffer has been set
<code>consp</code>	tests whether the value is a cons cell (in Lisp, the building blocks of lists)
<code>listp</code>	tests whether the value is a list
<code>numberp</code>	tests whether the value is numeric

Example:

```
(setq a "this is a string")
"this is a string"
(arrayp a)
t
(stringp a)
t
(numberp a)
nil
```

List functions

cons-cells
Lisp has numerous functions for processing lists. To understand these functions, it is important to understand the representation of lists in Lisp. Lists are formed by linking *cons cells*, which are cells consisting of two pointers. The first pointer, called `car`, references a value, and the second pointer, `cdr`, references the next cons cell of the next list element. The last cons cell of a list has a `cdr` value of `nil`. The following example depicts the representation of the list `(This is a list)` with cons cells.

nested lists
If `car` of one element points to additional cons cells, this produces a nested list. The following table introduces the most important list functions.

<code>car</code>	returns the first element of a list (the <code>car</code> of its first cons cell)
<code>cdr</code>	returns the rest of a list without its first element (the <code>cdr</code> of the first cons cell)
<code>cons</code>	creates a new cons cell

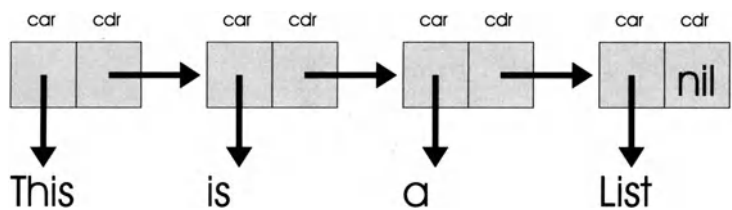


Figure 13.11. Internal structure of a list

nth	returns an element of a list
list	creates a new list
append	merges two lists lists by appending
reverse	inverts a list
sort	sorts a list

In addition to the normal Emacs Lisp functions, after loading a special extension, Common Lisp functions can be used as well. Two macros for list processing are `push` and `pop`, which add an element to the front of a list and remove one from there, respectively.

Example:

```
(setq l1 '(This is a list))
(This is a list)
(car l1)
The
(cdr l1)
(is a list)
(cons 'This (cdr l1))
(This is a list)
(append '(and this too) (cdr l1))
(and this too is a list)
(nth 2 l1)
a
```

String functions

The type string in Lisp is a specialization of the type array. This means that all functions that process arrays also work with strings. The following table presents the most important functions.

<code>make-string</code>	creates a string
<code>substring</code>	returns a substring
<code>concat</code>	concatenates strings
<code>string=</code>	compares two strings
<code>format</code>	formats a string similarly to <code>sprintf</code> in C

Example:

```
(setq a "this is a string")
"this is a string"
(stringp a)
t
(substring a 4 10)
" is a "
(make-string 10 32)
"
```

Cursor movement and text processing

inserting and deleting

The most important functions in Emacs Lisp are naturally those that process a text in a buffer, move the cursor, and insert or delete text. They allow automation of editor functions. The following table shows examples of such functions.

<code>point</code>	returns the current cursor position
<code>goto-char</code>	places the cursor at an absolute position
<code>forward-char</code>	moves the cursor one character to the right
<code>forward-word</code>	moves the cursor one word to the right
<code>search-forward</code>	searches for a string in the buffer
<code>insert</code>	inserts a string at the current cursor position in the buffer
<code>delete-region</code>	deletes the current region in the buffer

Interactive functions

Meta-x
arguments

To permit the user to invoke a function with a key binding or with **<Meta-x>**, this function must contain the function call `interactive`. This call is analogous to a declaration that specifies which arguments are to be passed to the function on invocation. The simplest case is a function without arguments, in which case it suffices to invoke `interactive` at the start of the function. The

following example shows such a simple function that could be bound to the key combination **<Shift-CursorRight>**, which would then simultaneously move the cursor one character to the right and mark the text.

```
(defun mark-move-right ()
  "move right and set the mark if it is not already active"
  (interactive)
  (if (not mark-active)
      (push-mark nil nil t))
  (forward-char))
```

It is possible to define a function that obtains its parameters via the user's minibuffer; this can also be done with the function `interactive`. As parameter a string containing a code character followed by a prompt is passed to `interactive`. The following example defines an interactive function that inserts a line composed of "-" characters into the text, whereby the length of the line is obtained first.

code characters

```
(defun line (len)
  "draw a -- line with length len"
  (interactive "nLength")
  (insert (make-string len 45)))
```

The code character used here is "n," meaning numeric input. If multiple parameters are passed, this is done in the same string but delimited by "\n." The following example shows a small extension of the above example that also allows specification of the character. The code character for inputting an individual character is "c."

inputting a number

```
(defun line (len lchar)
  "draw a line of lchar characters with length len"
  (interactive "nLength \ncChar")
  (insert (make-string len lchar)))
```

The complete description of all code characters can be obtained from the on-line help by typing **<Ctrl-h f>** `interactive` or from the Emacs Lisp Info files.

help

Notes on style

functional language

Lisp is a functional language, meaning that the most important structural element is the function. Pure procedures such as in Pascal, subroutines that do not return a value, do not exist in Lisp. Functional programming means whenever possible variable assignments should be avoided in favor of working directly with the return value of a function or the value of an expression.

variable assignments

13.8 Configuration

configuration file

`.emacs`

To customize the keyboard layout within Emacs or to load additional functions and packages, the file `.emacs` in the user's home directory is generally adapted. Emacs loads this file on starting, and the Emacs Lisp interpreter evaluates it. Thus, it is also called the *initialization file*.

site-lisp

Alternatively, such settings can be made globally for all users in the files `site-start.el` and `default.el`. They must reside in a directory that Emacs automatically searches for Lisp files, such as the directory `site-lisp`, which is usually located at `/usr/lib/emacs/site-lisp` under Linux. Other UNIX systems usually install Emacs under `/usr/local`. In this case the `site-lisp` directory would be at `/usr/local/lib/emacs/site-lisp`.

inhibit-default-init

If the file `site-start.el` exists, then it is evaluated by a user-specific initialization file. The file `default.el` is also optional and serves as the default initialization file. It is loaded after the user-specific initialization file unless the user's initialization file sets the variable `inhibit-default-init` to a value other than `nil`. This makes three configuration files, all of which are optional, which are invoked in the following order:

- `site-start.el` – for global settings made independently of the user-specific initialization file;
- `~/.emacs` – for user-specific settings;
- `default.el` – for global settings to be applied when the user does not disable them with the variable `inhibit-default-init`.

Configuration is carried out in these files with Emacs Lisp. The entries are Emacs Lisp function calls. As detailed in section 13.7, these are parenthetical expressions in which the first list element represents the function name and the remaining elements are the parameters. If the parameters are symbols, they are evaluated unless they are quoted with an apostrophe (`'`).

Emacs Lisp

Some of the paths of the configuration files contain the version number of Emacs. This number changes correspondingly with newer versions. In this section we describe version 19.28.

path

Beyond adapting the keyboard layout, the configuration files frequently modify global configuration variables and activate additional programs. A more advanced Emacs configuration file generally contains the following sections:

- Modification of global variables that toggle or change certain functions
- Settings for fonts and colors for Emacs under X11
- Loading of additional functions and packages
- Definition of commands whose invocation automatically loads the corresponding package or Lisp file
- Definition of hooks to be executed when a mode or a program starts
- Keyboard layouts

We now provide examples to explain these sections sequentially.

Global variables and settings

The following section of an `.emacs` file first loads the Common Lisp language extension (`cl`), which simplifies configuration. In addition, a small Lisp macro is defined as shorthand for the special form `condition-case` for ignoring errors. We do not discuss this macro further here. For detailed information on Lisp macros, refer to Lisp textbooks or the Info files for Emacs Lisp.

Common Lisp

ignoring errors

```
;; Load the Common Lisp language extension
(require 'cl)

;; macro to abbreviate condition-case
;; if all errors are to be ignored
```

```
(defmacro ignore-errors (&rest forms)
  "short form for a condition case"
  (list 'condition-case 'nil
        (cons 'progn forms)
        '(error nil)))

;; Path specification for Lisp programs
(push "/usr/lib/emacs/site-lisp/auctex" load-path)
(push "/usr/lib/emacs/site-lisp/swl" load-path)

;; Default mode for unknown file types
(setq default-major-mode 'text-mode)

;; Text width
(setq default-fill-column 78)

;; Make region visible
(setq transient-mark-mode t)

;; Permit eval with <Meta-Esc>
(put 'eval-expression 'disabled nil)
```

load-path The variable `load-path` contains a list of path specifications as strings. These paths are searched when an Emacs Lisp file is loaded. In a normal Linux installation where no modifications were made to Emacs, on startup this list at first contains an entry for `/usr/lib/emacs/19.28/lisp` of the official Emacs Lisp files and an entry for the Emacs `site-lisp` directory `/usr/lib/emacs/site-lisp`.

If additional Lisp program packages, such as the AUCTEX package, are installed in other directories, then this variable must be extended. Since the data structure is a list, the Common Lisp function `push` can be used to append an element to the front of the list.

default-major-mode The variable `default-major-mode` contains the name of the major mode that is to be used if no other mode can be found automatically for a file. The default is `fundamental-mode`. Since normal text files, in contrast to program files, usually cannot be distinguished by their name or extension, `text-mode` provides a reasonable alternative.

default fill column The variable `default-fill-column` permits specification of the column where word hyphenation can begin in a buffer with automatic hyphenation activated. Another such variable is `default-tab-width`, which enables setting the width of a tab mark in text.

The variable `transient-mark-mode` is a switch. If the variable contains a value other than `nil`, this deactivates the current selection of text, and thus the region, as soon as the buffer is modified. One side effect is that the current region cannot be highlighted when `transient-mark-mode` is set to `nil`. Thus the variable should always be set to `t`.

`transient-mark-mode`

The last line in the preceding section of code deviates somewhat from the other lines. It does not change a global variable, but reads the property list of a symbol. The key combination **<Meta-Esc>** or **<Esc-Esc>** actually represents the function `eval-expression`, which permits evaluating a Lisp expression directly in the minibuffer. Novices and former vi users who frequently need to terminate an accidental mode by pressing the **<Esc>** key would be too confused by this function. Therefore this feature is originally deactivated, and **<Meta-Esc>** produces a corresponding message. Users who are more familiar with Emacs and who know that the Escape key is not **<Esc>** but **<Ctrl-g>** would tend to be irritated by this message. By setting the property `disabled` to `nil`, the normal function of **<Meta-Esc>** is restored. The simplest way to obtain a description of a variable is by entering **<Ctrl-h v>**. After this entry, the minibuffer prompts for the name of the variable, and Emacs displays its description in a new window.

(put 'eval-expression
'disabled nil)

other global
variables & options

The function `edit-options` provides a very good overview of all global variables used as options in Emacs. In a new window it displays a list of all option variables along with their current values and their descriptions. The values can be modified directly in this window with simple keys, as Figure 13.12 shows.

Fonts and colors

Since the release of version 19, GNU Emacs under X11 can open its own window with pull-down menus, is mouse-driven, and displays various fonts and colors in a buffer. This section explains the configuration of fonts and colors, while mouse operation and pull-down menus are explained later together with keyboard layout.

Version 19

GNU Emacs manages various fonts and colors by means of *faces* (from “type faces”). A face is a specification of the font, the foreground color, the background color, and whether the font is

faces



Figure 13.12. The function edit-options

underlined. Faces defined in this way can then be assigned in the text area.

If the `transient-mark-mode` is active, then Emacs employs the face `region` to display the current region. Various major modes also use faces to display text. In Info mode, for example, references to other pages and options in menus are represented in special faces. For editing program files, the font-lock minor mode can be activated to display keywords, function names, comments, and strings with different faces.

Faces in Emacs Lisp are usually created with the functions `make-face` or `copy-face` and then modified with the functions `set-face-font`, `set-face-foreground`, `set-face-`

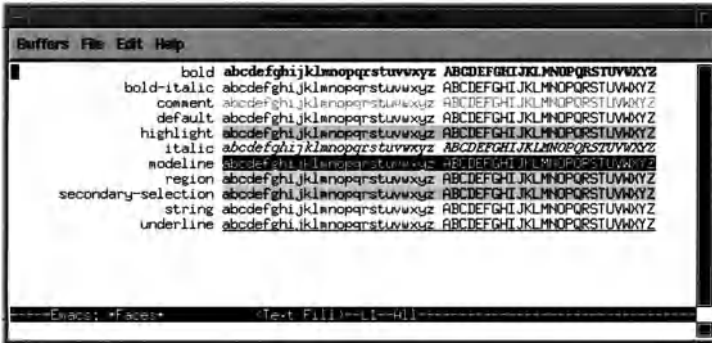


Figure 13.13. The output of list-faces-display

background, and set-face-underline-p. The function list-faces-display, whose output is shown in the following figure, provides an overview of the appearance of all defined faces.

overview

The following excerpt redefines several faces.

```
;;
;; fonts and faces
;; =====
;; Set fonts in Emacs
(when window-system
  (ignore-errors
    (set-face-font 'bold
      "-adobe-courier-bold-r-normal-*)")
    (set-face-font 'italic
      "-adobe-courier-medium-o-normal-*)")
    (set-face-font 'comment
      "-adobe-courier-medium-o-normal-*)")
    (set-face-font 'bold-italic
      "-adobe-courier-bold-o-normal-*)"))

;; Set faces for font-lock mode
(when window-system
  (setq font-lock-function-name-face 'bold)
  (setq font-lock-comment-face 'italic)
  (setq font-lock-string-face 'default)
  (setq font-lock-doc-string-face 'default))

;; Set attribute for faces
(when window-system
  (ignore-errors
    ;; Region
    (set-face-foreground 'region "black")
    (set-face-background 'region "grey90")
    (if (not (x-display-color-p))
      (set-face-underline-p 'region t))

    ;; bold face
    (set-face-foreground 'bold "black")
    (set-face-background 'bold "white")
    (set-face-underline-p 'bold nil)))
```

```
;; Additional faces make sense for color monitor
(when (and window-system (x-display-color-p))
  (ignore-errors
    (copy-face 'default 'comment)
    (set-face-foreground 'comment "grey60")
    (set-face-background 'comment "white")
    (setq font-lock-comment-face 'comment)

    (copy-face 'default 'string)
    (set-face-foreground 'string "gray10")
    (set-face-background 'string "white")
    (setq font-lock-string-face 'string)))
```

Loading / activating extensions

load Extensions are loaded with the Lisp function `load`. However, this function reports an error if the file to be loaded does not exist.

ignore-errors Therefore, we use the macro `ignore-errors`, which continues processing after an error. The package `func-menu` under X11 and in C mode can display a menu with all the functions defined in the current file. Selecting a function from this menu causes the cursor to jump to the definition of the function. This amounts to a simple but practical source code browser.

```
;; func-menu
(when window-system
  (ignore-errors
    (load "func-menu")
    (define-key global-map [S-down-mouse-3] 'function-menu)))
```

hooks Hooks are used to change options that affect individual modes. Hooks are variables provided by every mode, in which Lisp expressions can be stored. These expressions are executed at certain times, depending on the type of hook; normally this is the

initialization initialization of the mode for a new buffer. This means that certain functions can be executed each time a buffer is opened in a certain mode.

line numbers The following section of an Emacs initialization file uses this mechanism to activate the minor modes `line-numbers` and `font-lock` for buffers in C mode and in Emacs Lisp mode.

```
;; line-numbers and font-lock for c
(add-hook 'c-mode-hook
  (lambda ()
    (line-number-mode 1)
    (if window-system
      (font-lock-mode t))))
```

```
;; line-numbers and font-lock for emacs-lisp
(add-hook 'emacs-lisp-mode-hook
  (lambda ()
    (line-number-mode 1)
    (if window-system
      (font-lock-mode 1))))

;; line-numbers and font-lock for tcl
(add-hook 'tcl-mode-hook
  (lambda ()
    (line-number-mode 1)
    (if window-system
      (font-lock-mode t))))

;; Auto fill in text mode
(add-hook 'text-mode-hook
  (lambda () (turn-on-auto-fill)))
```

Emacs has numerous modes that are never activated without targeted configuration. To be used, they must be loaded using the `autoload` mechanism in Emacs, which allows loading specific Lisp files only when they are actually needed. The function `autoload` allows definition of such specifications. As parameters `autoload` receives a function name and the name of a file. When the function parameter is invoked, Emacs knows that the specified file must be loaded first.

To activate a new mode automatically when a file with a certain extension is edited, an entry must be made in the `auto-mode-alist`, a list containing pairwise entries of a pattern and a function. This specifies that the corresponding function is invoked when a file is loaded that matches the pattern. The following section establishes this correspondence for Ada files. With these settings, if a file is loaded whose name ends in `.ada`, then the function `ada-mode` is invoked. This function begins loading the file `ada.el`.

`auto-mode-alist`

Ada

```
;; Ada mode
(autoload 'ada-mode "ada"
  "Ada major mode." t)
(pushnew '("\\.ada$" . ada-mode)
  auto-mode-alist)

;; Smalltalk mode
(autoload 'smalltalk-mode "st.el" "" t)
(pushnew '("\\.st$" . smalltalk-mode)
  auto-mode-alist)

;; Modula-3 mode
(autoload 'modula-3-mode "modula3.el" "" t)
(pushnew '("\\.m3$" . modula-3-mode)
  auto-mode-alist)
(pushnew '("\\.i3$" . modula-3-mode)
  auto-mode-alist)
```

```
;; lispsdir -- Search for/retrieve additional
;; packages in the Emacs Lisp Archive dir
(autoload 'lisp-dir-apropos "lispsdir" nil t)
```

pushnew

The function `pushnew` is a Common Lisp macro that appends an element to the front of a list if the element is not already contained in the list. The `autoload` mechanism is also used in the above section for the function `lisp-dir-apropos`, which searches in the Lisp Code Repository (see section 13.6).

`lisp-dir-apropos`

function of a key

Next we define three small Lisp functions to which we will later bind keys. The function `mark-and-do` is used together with the shift key. **<Shift-ArrowRight>** moves the cursor to the right and simultaneously extends the marking, or sets it if it is not active. The function generally tries to detect the function of the pressed key without the shift key and to invoke this function after processing the mark. Here it uses the function `this-command-keys`, which returns key combinations that triggered the function.

```
(defun S-Key (keysym)
  "return the keysym without S-"
  (let ((name (symbol-name keysym)))
    (vector
     (make-symbol
      (substring name 2 (length name))))))

(defun mark-and-do ()
  "set the mark if not active and
do command without shift"
  (interactive)
  (if (not mark-active)
      (push-mark nil nil t))
  (let ((key (aref (this-command-keys) 0)))
    (call interactively
     (key-binding (S-Key key)))))

(defun mouse-describe-function (event)
  "describe function under the mouse-cursor"
  (interactive "e")
  (save-excursion
    (mouse-set-point event)
    (let ((fn (function-called-at-point)))
      (describe-function fn)
      nil)))
```

key bindings

The definition of key binding itself is relatively simple. The key and the function to be invoked are passed to the functions `define-key` and `global-set-key`. `define-key` is invoked again with the keymap where the binding is to be entered. The keymap

named `function-key-map` plays a particular role here. It replaces key sequences with other keys or symbols, before they can be processed by the other keymaps. Entries in the `function-key-map` are generally made to adapt terminals that send function keys as ASCII sequences. The codes for simple cursor keys are usually already in the `termcap` or `terminfo` database of the system and thus also known to Emacs.

function key-map

function keys

```
;; Special cursor Keys on Linux Console
(define-key function-key-map "\e[ " [home])
(define-key function-key-map "\e[4 " [end])
(define-key function-key-map "\e[2 " [insert])

;; Control-cursor
(global-set-key [C-right] 'forward-word)
(global-set-key [C-left] 'backward-word)
(global-set-key [C-prior] 'beginning-of-buffer)
(global-set-key [C-next] 'end-of-buffer)

;; Shift-cursor
(global-set-key [S-right] 'mark-and-do)
(global-set-key [S-left] 'mark-and-do)
(global-set-key [S-up] 'mark-and-do)
(global-set-key [S-down] 'mark-and-do)
(global-set-key [S-end] 'mark-and-do)
(global-set-key [S-prior] 'mark-and-do)
(global-set-key [S-next] 'mark-and-do)

;; Function Keys
(global-set-key [f1] 'info)
(global-set-key [f2] 'save-buffer)
(global-set-key [f3] 'find-file)

(global-set-key [f5] 'goto-line)
(global-set-key [S-f5] 'what-line)
(global-set-key [f6] 'tags-search)
(global-set-key [S-f6] 'visit-tags-table)

(global-set-key [f9] 'compile)
(global-set-key [M-f8] 'next-error)
(global-set-key [f10] 'next-error)
(global-set-key [f12] 'advertised-undo)

;; Redefine home and end
(global-set-key [home] 'beginning-of-line)
(global-set-key [end] 'end-of-line)

;; Redefine backspace and delete
(define-key function-key-map [delete] [deletechar])
(define-key function-key-map [backspace] [DEL])
(global-set-key [DEL] 'delete-backward-char)

;; divers
(define-key ctl-x-map "p" 'toggle-save-place)
(define-key lisp interaction-mode-map [M-return]
  'eval-print-last-sexp)

(define-key emacs-lisp-mode-map [S-mouse-1]
  'mouse-describe-function)
(define-key lisp interaction-mode-map [S-mouse-1]
  'mouse-describe-function)
```

menus

A new menu is defined like a key binding and stored along with the normal key bindings. The following example defines a local menu for Emacs Lisp mode.

```
(when window system
  ;; New menu in elisp mode
  (define-key emacs-lisp-mode-map [menu-bar elisp]
    (cons "Elisp" (make-sparse-keymap "elisp")))
  (define-key emacs-lisp-mode-map [menu-bar elisp debononoff]
    '("Cancel Debug on Entry" . cancel-debug-on-entry))
  (define-key emacs-lisp-mode-map [menu-bar elisp debonen]
    '("Debug on Entry" . debug-on-entry))
  (define-key emacs-lisp-mode-map [menu-bar elisp debdefun]
    '("Debug Defun" . edebug-defun))
  (define-key emacs-lisp-mode-map [menu-bar elisp evalbuff]
    '("Eval buffer" . eval-buffer))
  (define-key emacs-lisp-mode-map [menu-bar elisp evalreg]
    '("Eval Region" . eval-region))
  (define-key emacs-lisp-mode-map [menu-bar elisp evaldef]
    '("Eval Defun" . eval-defun)))
```

define-key

text and action

The first parameter of `define-key` contains the keymap, the second the position in the menu hierarchy. `[menu-bar elisp]` defines a new pull-down menu. The third parameter defines the text and the associated action. In the first invocation of `define-key`, the last parameter does not define any actual action, but a keymap for the new pull-down menu. The other invocations each contain a cons cell whose first element specifies the text for the menu and the second element the name of the function to be invoked.

Languages & Tools

One particular feature of Linux is the large number of freeware programming languages in binary format available from Linux FTP servers and contained in the various Linux distributions. These freeware tools seldom take a back seat to proprietary systems. This chapter introduces the various compilers, interpreters, and software development tools that are available under Linux and identifies their most important features.

FTP servers

14.1 Languages

Undisputedly, C holds the position of the most important programming language for UNIX systems. Today's UNIX itself and all its utilities were developed largely in C. So it is not surprising that a C compiler serves as a central element of a UNIX development environment.

C and UNIX

As a rule, proprietary UNIX systems of the past have included a C compiler in their packages. However, this has changed recently. Newer PC UNIX implementations are now being marketed in user versions without a C compiler and without network support. The full version including these components costs significantly more. Programming languages supported under Linux range from C, C++, and Objective C to Lisp and Prolog and on to Smalltalk and Forth (see Figure 14.1). Classical languages like Fortran and APL are also available under Linux. Even Basic programs can be developed with either of two interpreters.

C compiler

C++, Objective
C, Lisp, Prolog,
Smalltalk, Forth,
Fortran, APL, Basic

Compilers are available for Modula-2 and Modula-3 as well as for Oberon (see Figure 14.2). Some of these compilers are commercial systems that are also available for other UNIX systems.

Modula
Oberon

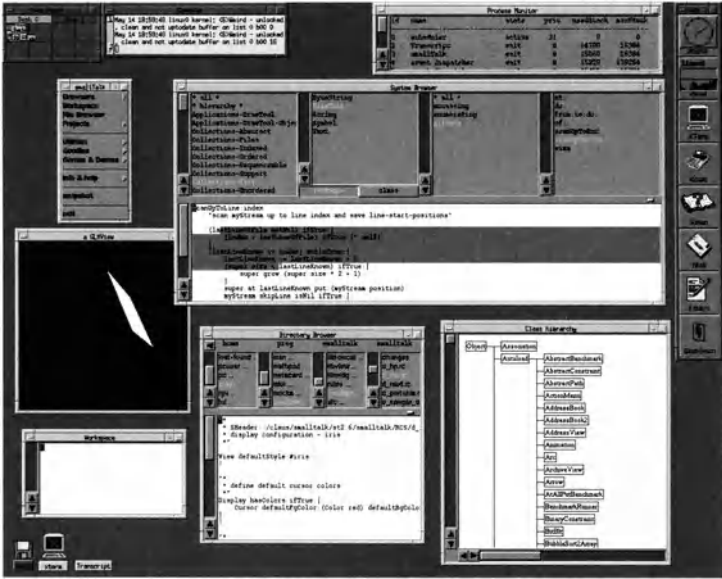


Figure 14.1. Smalltalk/X under Linux

GNAT, Ada9X GNAT, the Ada translator for Ada9X developed at New York University, is also available for Linux. It offers a complete compiler that uses the GCC back end. Since it became available before the official completion of the Ada9X standard, it has not been granted official validation just yet.

In view of recent developments, in the near future we can anticipate that almost every language will be available under Linux.

14.2 C compilers

gcc The C compiler used under Linux is the GNU C Compiler (gcc) of the Free Software Foundation, which can generate optimized code. gcc's porting availability for numerous UNIX platforms facilitates the porting of software. The GNU C Compiler supports ANSI and K&R standards for C as well as C++ and Objective C. This compiler's detailed error messages on syntax errors deserve special note.

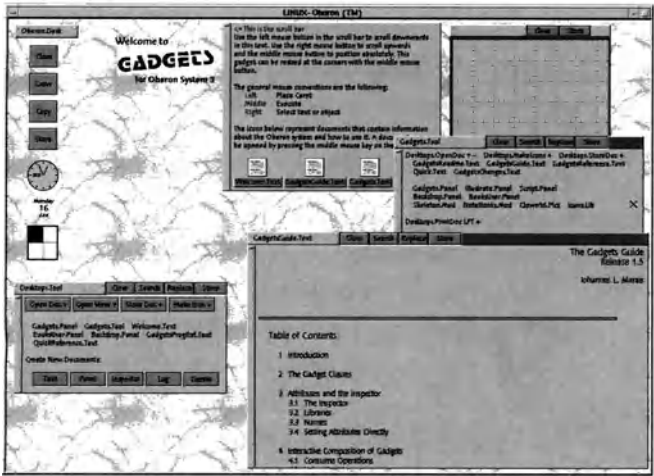


Figure 14.2. Oberon System 3 under Linux

14.3 Pascal, Fortran, Simula, and Modula-2

The GNU C compiler also provides the basis for the Pascal-, Simula-, and Fortran-to-C converters. These converters read existing Pascal, Simula, or Fortran source code and generate a C program, which is then compiled with the C compiler. Skillful integration in the make system can leave the intermediate stage in C completely unnoticed by the programmer.

converters

Beyond standard Pascal, the Pascal-to-C converter can even translate dialects of several Pascal variants, such as Turbo Pascal up to Version 5 and Macintosh Pascal. The Simula-to-C converter works according to the same principle. The German Association for Mathematics and Data Processing (Gesellschaft für Mathematik und Datenverarbeitung, GMD) has ported its Modula-2 development environment MOCKA to Linux. It is interesting that the compiler itself was implemented in Modula-2; its source code is included in the package. In contrast to the above tools, MOCKA generates object code directly. MOCKA also supports linking C routines as *foreign modules*.

Pascal

Simula

Modula-2

14.4 Lisp and Prolog

Since the programming languages used in the area of knowledge-based systems and artificial intelligence play an important role at universities, several implementations have become available. Here Lisp is of particular importance because it achieved relatively broad propagation as the programming language of the GNU Emacs.

AI
Emacs Lisp

One Common Lisp implementation with an object-oriented extension (a subset of CLOS) named `clisp` is included in most Linux distributions. The extensive GNU Common Lisp is based on the previous `akcl` and orients itself to the new ANSI standard for Common Lisp.

clisp

The FTP server `swi.psy.uva.nl` contains the extensive Prolog implementation of the University of Amsterdam, which is easy to compile under Linux. It is based on the Warren Abstract Machine (WAM) and provides practical on-line help. An interesting aspect is the connection to XPCE, an object-oriented system that also comes from Amsterdam. This system provides a graphical programming environment for Prolog.

Prolog

XPCE

14.5 Tcl

Whether editor, database, or modem program, good software needs an extension language with which the user can write scripts or macros. These can be used for automation of sequences or for simple extension of the system. Well-known examples of such languages include Emacs Lisp, `elk`, and Microsoft Visual Basic for Applications (VBA).

extension language

Emacs Lisp, `elk`

At the end of the 1980s, John Ousterhout conceived Tcl (Tool Command Language) as such a language. Tcl has an uncomplicated syntax and contains a single simple data type, the character string. Depending on requirements and context, these strings can be interpreted as numbers, lists, or other data types. The power of the language stems mainly from its many high-level functions and the fact that, similar to Lisp, it does not distinguish between data and program. Tcl code itself is stored and interpreted as strings. This makes it easy

strings

high-level functions

to realize an extension of language constructs or customized code.

The language was implemented as a C library and can be combined with existing programs by linking and executing a few functions. For independent programs there are directly executable interpreter s that read a single input, pass it to the Tcl interpreter of the Tcl library, and then output the result. Here we use small examples to introduce some of the most important language elements of Tcl. A Tcl interpreter can be started by entering `tcl` or `tclsh`.

C library

interpreter

The language

In Tcl a list is a string whose elements are separated by blanks, tabs, or line breaks. A Tcl statement is a list, i.e., likewise a string, whereby the first element is the name of a function to be invoked. The remaining elements are passed as arguments to the function.

lists

statements

```
hermes:/home/strobel> tclsh
% set a 5
5
% puts $a
5
% puts "The value of a is $a"
The value of a is 5
```

In the above example the Tcl interpreter first evaluates the line `set a 5`. The function `set` assigns a value to a variable, here `a`. A variable in Tcl need not be declared; it exists as soon as a value is assigned to it. The next line outputs the current value of `a`. Before the Tcl interpreter evaluates a line of Tcl code, substitutions are made that are induced by special characters. The `$` character, e.g., causes the subsequent name to be handled as a variable and replaces it with its value. If there is no variable with the specified name, there is an error.

variables

This variable evaluation mechanism works anywhere in a program line unless it is prevented by special quoting. The following example shows that even the function name is no exception.

evaluation

```
% set fkt puts
puts
% set arg "The function $fkt was invoked here"
The function $fkt was invoked here
```

```
% $fkt $arg
The function $fkt was invoked here
%
```

puts In the last line the function `puts` is invoked. Its argument is the string "The function `$fkt` was invoked here."

function call Another type of evaluation is invoked by the square brackets `[]` return value. The text between the brackets is evaluated as a Tcl function invocation and then replaced by the returned function value. The following example uses the function `lindex`, which interprets a string as a list and returns a certain element. The numbering of list elements in Tcl begins with the ordinal number 0.

```
% set l "This is a list"
This is a list
% puts "The 2nd element is: [lindex $l 1]"
The 2nd element is: is
%
```

associative arrays Another interesting aspect of Tcl is arrays, whose indices are not restricted to numeric values. Arbitrary strings are permitted. These are called associative arrays. The next example uses `grade` as an array and the names of fictional students as indices.

```
% set grade(sam) 2.3
2.3
% set grade(fred) 1
1
% set grade(peter) 4
4
% puts $grade(fred)
1
% set name sam
hugo
% puts "The grade average for $name is $grade($name)"
The grade average for sam ist 2.3
%
```

control constructs Tcl provides the usual control constructs of procedural languages. In addition to `if`, `switch`, `for`, and `while`, there is a `foreach` loop that executes a block for each element of a list. This permits iteration over the elements of an associative array, as the following example shows:

```
% set students [array names grade]
peter fritz hugo
% foreach name $students {
    puts "The grade average for $name is $grade($name)"
}
The grade average for peter is 4
The grade average for fred is 1
The grade average for sam is 2.3
%
```

The function `array` returns information about existing arrays. With the argument `names` all indices are returned for which the array contains a value.

Tcl also provides functions for working with regular expressions (see section 2.8). The function `regexp` compares a regular expression with a string and returns 1 if the expression was found. Optionally, substrings that match the specified expression or its subexpressions can be assigned to variables.

regular expressions

```
% set s "123abchello6677pplist7zz"
123abc123hello6677pplist7zz
% set pattern {hello.*7([a-z]+)7}
hallo.*7([a-z]+)7
% regexp $pattern $s all sub1
1
% puts $all
hallo6677pplist7
% puts $sub1
pplist
%
```

The curly braces `{}` have a function similar to that of the double quotation marks. However, they prevent further internal substitution. The regular expression in the above example had to be written in curly braces because otherwise the substring `[a-z]` would have been interpreted as a function call.

curly braces

Custom functions can be defined in Tcl by invoking the function `proc` `proc`. The arguments of the new function can be defined as individual variables or as list variables. In the former case the number of parameters is fixed, while it is dynamic in the latter case.

custom functions

```
% proc myFunction {arg1 arg2} {
    puts "The arguments are $arg1 and $arg2"
    return 1
}
% myFunction a b
The arguments are a and b
1
```

```
% proc funct2 args {  
    puts "The arguments are $args"  
    return "bye"  
}  
% funct2 a b c d e  
The arguments are a b c d e  
bye
```

local variables Variables in a Tcl function are normally local. To allow access to global variables, they must first be specified in a `global` line:

```
% set name Sam  
Sam  
% proc Test {name} {  
    puts "name is $name"  
}  
% Test  
can't read "name": no such variable  
  
% proc Test {} {  
    global name  
    puts "name is $name"  
}  
% Test  
name is Sam  
%
```

Tcl scripts To enable using Tcl scripts as executable files, the Tcl interpreter is specified like a shell in the first line after `#!`. If users have execution permissions to the script file, then they can launch the script by entering its name. The following example shows the start of such a script file:

```
#!/usr/bin/tcl  
puts "Hello world!"  
...
```

Applications and extensions

There are two basic approaches to writing applications with Tcl. First, an application written in C can be extended by a script language with the Tcl interpreter. Linking the script language to the application occurs via the definition of additional Tcl commands. This approach makes sense primarily with existing programs. For a new development, even the main application could be developed in Tcl. Functions that are not available can be realized as Tcl extensions in C and accessed via Tcl.

script language

Tcl extensions

Extending Tcl is very simple. You define a C function, which, just as a main function in C, receives its parameters via `argc` and `argv`. Then in the initialization mode of the Tcl interpreter you invoke the function `Tcl_CreateCommand`, which registers the new function and assigns it a name within Tcl.

Many Tcl language extensions are available in the public domain. These extensions enable access to SQL databases or UNIX system invocations. For example, *Tcl-dp* provides functions for network programming (sockets and RPC) and *itcl* extends Tcl by classes and methods for object-oriented programming. A debugger and a class browser are also available.

In addition to TCP/IP sockets and Sun RPCs, the extension *scotty* also provides functions for accessing DNS and for communication via SNMP. The following example shows a simple server that was written with *scotty*. It opens a TCP socket and waits for connections. The Telnet program is one option for a client. When a client connects to the server, the server sends the current date and time to the client and waits for a reply from the client.

`argc, argv`

SQL

sockets

OOP

DNS and SNMP

TCP socket

```
#!/usr/local/bin/scotty -f

set port 1371

proc handleConnection {lsock} {
    set socket [tcp accept $lsock]
    # send date and time to client
    puts $socket [exec date]
    # wait for reply from client
    puts "Msg from Client : [gets $socket]"
    # end connection
    close $socket
}

set lsocket [tcp listen $port]
addinput -read $lsocket "handleConnection %F"
puts "waiting for connections on port $port ..."
```

The client's output could look like this:

```
hermes:/home/strobel> telnet hermes 1371
Trying 194.45.197.100...
Connected to hermes.stud.fh-heilbronn.de.
Escape character is '^]'.
Fri Jan 6 16:16:16 MET 1995
hallo
Connection closed by foreign host.
hermes:/home/strobel>
```

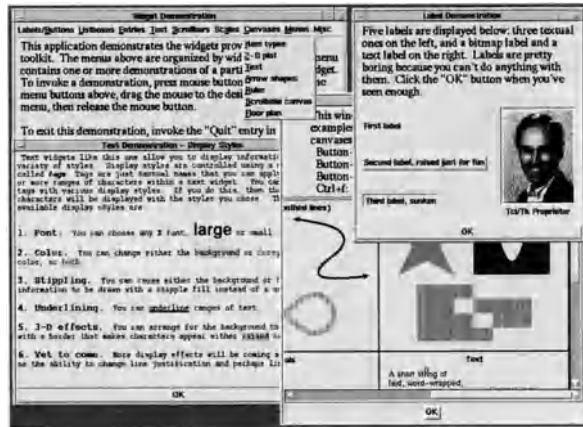


Figure 14.3. The Tcl/Tk interface builder XF

Tk

widget set Tk is a widget set that has interfaces to both C and Tcl. Thus it proves especially suitable for providing Tcl programs with a graphical user interface. Due to the underlying interpreter language, simple and prototypes rapid prototypes as well as complete applications can be developed.

Prominent examples of Tcl/Tk applications include Zircon, tkined, Picasso, and many other systems that employ Tcl as their extension language.

XF

interface builder The interface builder XF for Tcl/Tk enables graphical and interactive design of a Tk user interface (see Figure 14.3). This not only makes implementing Tk interfaces simple, but allows subsequent loading of existing Tcl/Tk programs and modifying or extending them. Since XF itself was developed in Tcl/Tk, the user interface can be tested in the course of development, and the representation during development corresponds exactly to the finished program.

Tcl/Motif

OSF/Motif To couple the advantages of Tcl with the Motif widget set, try Tcl/Motif. This package uses normal Motif widgets to generate graphical user interfaces. Thus Motif user interfaces can be designed

simply in an interpreter environment. The programmer can employ almost all of the accustomed widget resources. Tcl/Motif even supports the drag and drop mechanism of Motif 1.2.

resources

An FTP server containing numerous Tcl extensions as well as finished Tcl/Tk applications is `ftp.aud.alcatel.com`. The following WWW home page provides an overview of Tcl extensions, documents and programs:

WWW

`http://web.cs.ualberta.ca/~wade/Auto/Tcl.html`

14.6 Interface builders

The most comfortable way to produce X11-based applications is with special software for the interactive design of graphical user interfaces. These interface builders can usually generate C source code, which the programmer can extend as needed. ParcPlace markets such an interface builder, called ObjectBuilder (see Figure 14.4). It permits direct manipulation of graphic objects with the mouse and generates C++ source code. Together with the ObjectLibrary, user interfaces can be generated with the OpenLook or Motif convention. Command-line options determine the look and feel. Both products are available for all common UNIX Platforms, with a purchase price of over \$1,000 U.S. dollars. Only the Linux version is available free and may be copied freely. For serious use, however, the user manuals, available directly from ParcPlace, should be at hand.

interactive design

ObjectBuilder

ObjectLibrary

VXP

Although the free VXP interface builder by Young Chen is still under development, it easily suffices for creating simple OSF/Motif-based user interfaces (see Figure 14.5). Contrary to the interface builder by ParcPlace, VXP generates pure Motif source code. All graphic objects can be positioned and manipulated by mouse. VXP then generates the corresponding C source code and an appropriate makefile. The functionality of the application has been implemented, as usual under Motif, within callback routines in C. VXP also manages the source code that the programmer adds. In addition, the C compiler can be started with a click of the mouse without leaving

OSF/Motif

makefiles

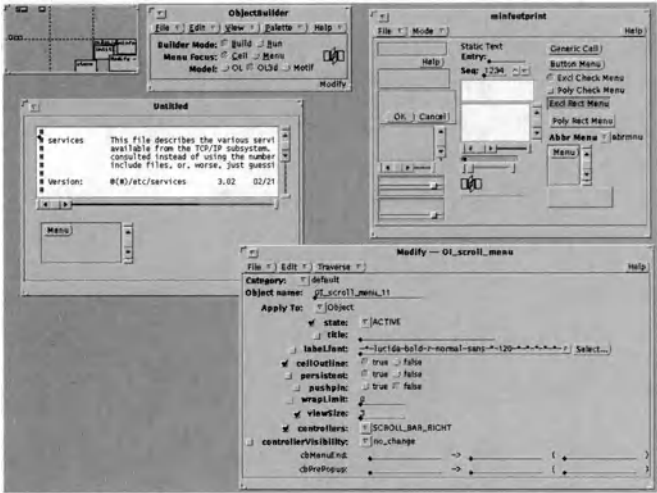


Figure 14.4. The interface builder by ParcPlace



Figure 14.5. VXP Motif interface builder

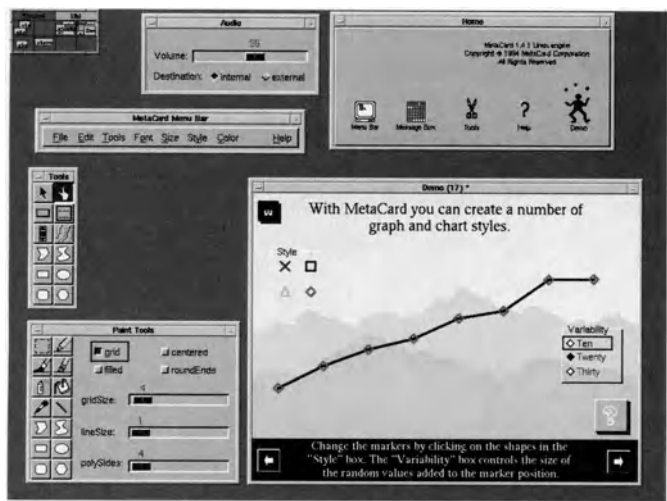


Figure 14.6. Metacard under Linux

the development environment. The source code generated by VXP needs no library routines other than the Motif library, making it easily portable to other UNIX platforms.

14.7 Metacard

Metacard is a system for developing Hypercard applications, e.g., as we know them from the Apple Macintosh. Fortunately, Metacard is able to load and process Apple Hypercard stacks. The integrated programming language permits easily developing smaller applications or prototypes for graphical user interfaces. Metacard is available on all common UNIX platforms, but it is commercial software (see Figure 14.6).

Hypercard
Apple

14.8 awk, gawk

awk is a traditional UNIX tool for the production of smaller scripts for processing strings and text files. The name of the program derives from the initials of its authors: Aho, Weinberger, and Kernighan.

processing text files

The strong resemblance of `awk`'s interpreter language to C makes it relatively easy to learn.

`awk` does not distinguish between numeric and string variables and requires no variable declarations. For smaller projects, `awk` can serve as a prototyping tool. However, the size of an `awk` program should not exceed circa 200 lines of code.

The following script totals the sizes of all files in the current directory and displays the sum:

```
linux1:/> ls -l | awk '{sum += $5} END {print "Summe : " sum}'
```

The original utility is not available under Linux, but the GNU implementation of the Free Software Foundation called GNU `awk` (`gawk`) is. In addition to `awk`, commercial UNIX systems often contain the extended version called `nawk`.

14.9 Perl

Perl is also an interpreter language. It combines the most important features of `sed`, `awk`, and the usual UNIX shells and proves especially suitable for processing text files. One advantage over classical tools is the possibility of opening multiple files simultaneously.

Perl permits the use of lists and associative arrays, with the size of the data structures limited only by available memory, which allows the processing of relatively large amounts of data. Naturally, Perl also has loops and other control structures, subroutines, recursion, and regular expressions. Various format statements for data output enable the creation of clear reports and tables. An integrated debugger allows the setting of breakpoints and stepwise execution of a program.

Perl proves suitable for system administration because it allows the production of scripts that run with root permission. It is also noteworthy that almost all routines of the standard C library and the UNIX kernels can be used directly under Perl, including functions for network programming via sockets. The new version 5 of Perl offers object-oriented extensions.



Figure 14.7. Software development with Emacs

14.10 Editors

A compiler alone does not constitute a full-scale development environment. At least a befitting editor and a symbolic debugger are required as well. The GNU Emacs editor integrates these components (see Figure 14.7). It can invoke numerous utilities like `grep` and `RCS` as well as the C compiler and the GNU debugger, so that the complete development cycle can run within the editor. Emacs is described in detail in Chapter 13.

Besides the GNU Emacs, Linux distributions furnish numerous other editors, which are introduced in Section 12.2.

14.11 GNU Debugger (GDB)

The GNU Debugger (GDB) is a powerful symbolic debugger that supports C, C++, and Modula-2. It provides all functionality that developers expect from such a tool. Programs can be executed stepwise. Breakpoints allow the interruption of the execution at defined points, and watchpoints interrupt the program when specified values are modified. An option allows the continuous display of the values of selected variables or objects.

Newer versions of GDB also permit remote debugging, which means that the debugger can run on a different machine from the program to be debugged. The connection between the machines

Emacs

multiple languages

breakpoints

watchpoints

remote debugging

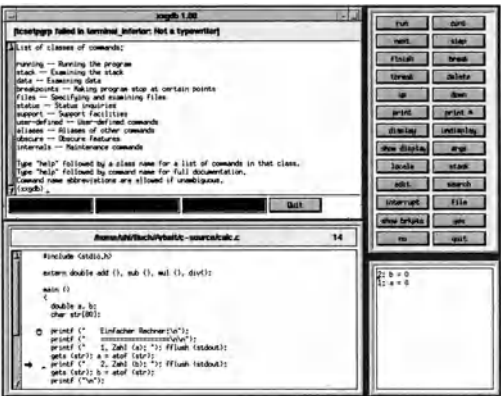


Figure 14.8. The GDB front end xgdb

can be either a serial interface or a network. Under Linux, even the operating system can be analyzed with GDB, and running processes can be debugged.

The GNU Debugger provides only a simple command-line-oriented user interface. However, there are graphical front ends that enable comfortable mouse-driven operation of the debugger. These front ends start GDB as a second process and redirect its input and output. Thus they simulate user input and redirect debugger output to various windows. One example of such a graphical front end is xgdb (see Figure 14.8).

The top pane displays the current position in the source code. The central area contains a number of command buttons that invoke basic debugger functions. The second pane from the bottom allows the user to enter textual commands when keyboard input seems more practical.

GDB permits the continuous display of variable values in the bottom pane. An expression that occurs in the C source code can be evaluated easily: select the expression in the source code pane with the mouse; then click on the print button, and the result is displayed in the bottom pane.

You can define a breakpoint similarly by positioning the text cursor at the respective position in the source code pane and pressing the corresponding button. Breakpoints are displayed with the symbol



Figure 14.9. GDB in Emacs

of a flat hand; a blue arrow marks the current position in the source code.

Another way to make the use of the GNU Debugger more comfortable is by integrating it with the Emacs editor. A special mode in Emacs permits debugging programs within a normal editor window, which is divided into two panes. The upper pane displays the source code and marks the current position with an arrow. The lower pane controls the debugger and allows input of commands.

Emacs

Using GDB within Emacs has some advantages for Emacs specialists, but it is certainly not as comfortable as using an X-based front end.

for experts

14.12 Make utility

Another important component of the C development environment is the `make` utility. This command significantly simplifies the compilation of a project that consists of multiple modules. For this purpose, the programmer stores the dependencies between the individual program parts in a makefile. If the programmer modifies the source code of one of these modules, then only the modified parts and the modules dependent on them are recompiled.

makefile

The GNU variant of `make` used under Linux has several options that go beyond the normal `make` utility. For example, it supports the Revision Control System (RCS), a collection of commands for

GNU make

RCS

version management of source code. If a file cannot be found in the current directory, GNU `make` seeks a subdirectory named `RCS` from which it reads the newest version of a module automatically. This even applies to the makefile.

implicit rules

GNU `make` also recognizes more implicit rules than other versions of `make`. A makefile for a small C program consisting of three files, for example, could take the following form:

```
CFLAGS = -g
LDLIBS = -lm

test: test.o sub1.o sub2.o
```

linking

GNU `make` contains rules to create `.o` files from the corresponding `.c` files and to link the program `test` from all object files and with the specified libraries. The documentation for GNU `make`, which is contained in the Info system, provides an overview of all rules and variables (also see section 11.2).

14.13 Imake

platform-independence

The differences between various UNIX variants often seriously encumber platform-independent software development. Hence it might become necessary to produce different versions of makefiles, each tailored to a respective system. Here the `Imake` utility known from the X Windows System provides support by generating a platform-specific makefile from a platform-independent `Imakefile`.

xmkmf

`Imake` is normally invoked in the script `xmkmf`. For perfect functioning, this command requires *template files* that contain the system-specific data. These files are installed in the directory `/usr/lib/X11/config`.

14.14 RCS

version management

The development of larger projects – consisting of a multitude of different modules and having new versions thereof evolving steadily – makes a version-management system indispensable. Such a system

gains in importance when multiple programmers are working on a project simultaneously.

Since all versions of a program in its development cycle are archived, if the need arises, a programmer can return to the source code of a long-since-outdated version. In addition to the Source Code Control System (SCCS) known from UNIX System V, Linux offers the more powerful Revision Control System (RCS). Normally the archive data are stored in a separate directory named `RCS` or `SCCS`.

RCS provides numerous commands, the most important of which we outline here.

- `ci` (check in) transfers the specified file into the `RCS` directory and thus freezes the current version. The complete version is not stored with each check-in; only the difference with respect to the predecessor version, as determined with the `diff` command, is stored. The version number of the module is automatically incremented. check in
- `co` (check out) creates a nonmodifiable copy of the current version or of any predecessor of a module. If a module is to be modified, this must be indicated with the option `-l`. This protects the file from simultaneous modification by another programmer, for it can be checked out only once by only one user. check out
- `rlog` displays various information such as the status of an archive and the versions of the modules it contains. status

Other commands permit, for example, merging two versions. To allow the version management and the rest of the development environment to work together smoothly, newer versions of the Emacs editor have a special mode for operating RCS. Emacs

14.15 xwpe

`xwpe` poses an interesting alternative to Emacs as a development environment. Users who are familiar with Borland products for DOS (Turbo C, Turbo Pascal, etc.) will soon feel at home, for `xwpe` greatly resembles them (see Figure 14.10). Although `xwpe` is a text-oriented application, comfortable mouse operation is possible. `xwpe`'s author Borland
mouse



Figure 14.10. xwpe as a development environment

also developed a terminal emulation program with color support for the X Window System. However, xwpe also runs on a normal terminal, although its look suffers considerably.

The integrated editor permits processing multiple texts in different windows. Both the compiler (gcc) and the debugger (gdb) can be started with either the keyboard or the mouse. Error messages are displayed in a separate window, and breakpoints can be defined directly in the editor.

xwpe can be drawn from the FTP server `ftp.rzrn.uni-hannover.de` in the directory `/pub/systems/unix/xwpe`.

14.16 Example

To give the reader a better impression of working with the previous development environments, we introduce a small C program. The program consists of multiple modules to add, subtract, multiply, and divide two numbers that are input. The main program contains only the two input prompts and the output of the results. The computations take place in separate modules. (Naturally, this simple example could have been combined in a single file.)

To input the first file, invoke the Emacs editor with the file `main.c` as argument. Since the file does not exist, Emacs starts with an empty buffer, which is already in C mode, however. After

the following source text has been input, it is saved with the key combination **<Ctrl-x> <Ctrl-s>**. This function could have been invoked from the menu instead.

```
#include <stdio.h>

extern double add (), sub (), mul (), div();

main ()
{
    double a, b;
    char str[80];

    printf ("    Simple Calculator:\n");
    printf ("    =====\n\n");
    printf ("    1.  Zahl (a): "); fflush (stdout);
    gets (str); a = atof (str);
    printf ("    2.  Zahl (b): "); fflush (stdout);
    gets (str); b = atof (str);
    printf ("\n");

    printf ("    a + b: %f\n," add (a, b));
    printf ("    a - b: %f\n," sub (a, b));
    printf ("    a * b: %f\n," mul (a, b));
    printf ("    a / b: %f\n," div (a, b));

    printf ("\n    Goodbye!\n");
}
```

The file `calc.c`

Now the other modules can be entered. The key combination **<Ctrl-x> <Ctrl-f>** opens a new file.

new files

```
double add (double a, double b)
{
    return a + b;
}

double sub (double a, double b)
{
    return a - b;
}
```

The file `addsub.c`

```
double mul (double a, double b)
{
    return a * a;
}

double div (double a, double b)
{
    return a / b
}
```

The file `muldiv.c`

Now the project consists of three files. This also has an effect on the makefile:

```
CFLAGS = -g
OBJS = calc.o addsub.o muldiv.o

calc: $(OBJS)
      $(CC) -o $@ $(OBJS)
```

The makefile

CFLAGS	First the environment variable <code>CFLAGS</code> is set with the option <code>-g</code> , so that the compiler generates debug code. The newly defined
OBJS	variable <code>OBJS</code> contains the names of all object files after their declaration. This declaration serves only to increase the readability
target	and is not absolutely necessary. The target is defined as the compiled and linked program <code>calc</code> . Generating the target requires three object files. The following command links these object files after their compilation:

```
$(CC) -o $@ $(OBJS)
```

rules	No separate rule needs to be specified for the translation of a <code>*.c</code> file to a <code>*.o</code> file. This is implicit in the make mechanism. On entry of
tab	the makefile, note that a <Tab> must precede the link command. After
compile	the makefile has been saved, the complete project can be compiled and linked from within the editor using the command <Meta-x> compile . The editor window splits. The lower window pane logs the commands that are executed and displays any error messages that arise. In our case we deliberately omitted a semicolon in the function <code>div</code> of the module <code>muldiv.c</code> . The compiler will detect this syntax error when it compiles the source code.

next error	The command <Meta-x> next-error allows the processing of the error in the editor and places the cursor at the position in the source code where the error occurred. After corrections, entering the command <Meta-x> compile starts the compiler again. Naturally, the programmer should assign each of the above commands to a keystroke combination (shortcut) to keep from having to enter the
------------	---

complete command each time. Such definitions are best placed in the file `.emacs` in the home directory (also see Chapter 13).

If logical errors have found their way into the program, the GNU Debugger provides useful support in detecting them. Since the program was compiled with the compiler option `-g`, the debugger can be started immediately. debugger

```
linux2:home/tul> xxgdb calc
```

First the required breakpoints need to be set. This can be done with the mouse or by entering the appropriate command. The cursor is placed in the respective line in the text window where a breakpoint is to be set. A click on the break button of the debugger then sets the breakpoint. Sometimes it is easier to set a breakpoint via the command line, especially if the breakpoint is to be at the very start of a function. breakpoints

```
gdb> break main
```

The preceding command stops the program as soon as it reaches `main`.

After all breakpoints have been defined, the program can be started with the `run` command. When execution stops at a breakpoint, it can be continued stepwise and the values of variables can be displayed. run

```
gdb> print a
```

The above command displays the value of the variable `a` if it is within the current scope. The command `display` initiates the continuous display of variable values.

```
gdb> display b
```

The preceding command displays the value of variable `b` after each successive `gdb` command.

The debugged program can be started directly from Emacs. However, first a shell must be opened with the command **<Meta-x> shell**. Then all UNIX commands can be used within the editor. The advantages of a shell that is executed within Emacs are the possibility to easily edit commands that have been executed and the ability to apply all editor commands, such as copying blocks, to the shell output.

14.17 Porting software

Only a small percentage of the programs that are available on the many FTP servers for UNIX systems have been especially ported to Linux. Since Linux adheres to the most important UNIX standards, however, compiling programs written for other UNIX systems generally poses no problem.

Unpacking

The programs on FTP servers have usually been stored as compressed and packed `tar` files. Such files were created with the `tar` program and can contain multiple files and directories. The name of the file usually contains information about its nature. If the name ends with `.tar.Z`, it is a `tar` archive that was compressed with the UNIX `compress` command. The extension `.tar.gz`, `.tgz`, or `.taz` indicates that it is a `tar` archive that was compressed with the `gzip` program.

To decompress and unpack such a file under Linux, it suffices to enter the following:

```
linux1:/home/tul> tar xvfz <file.tar.Z>
```

The `tar` program independently invokes the `gzip` program, which decompresses files that were compressed either with `gzip` itself or with the UNIX `compress` command.

Source codes that are sent as e-mail are often in shell archives whose names normally end with `.shar`. A normal Bourne shell suffices to extract the files from such an archive:

```
linux1:/home/tul> sh <file.shar>
```

Documentation

After unpacking an archive, the user should definitely read any included `README` or `install` files, which contain important instructions for compilation. These usually address the supported platforms and any problems that might arise.

`README`

Packages often contain extensive user documentation in the form of a PostScript or `TEX` file. At least a UNIX on-line reference Manual page should be included in any case.

`TEX` files

Makefile/Imakefile

The source code of a program is always accompanied by a makefile in which important options can be set, including compiler options, library paths, and the installation directory. The makefile also aids in the compilation and installation of the program. The invocation of `make` and `make install` starts the compiler and the linker, respectively, and then installs the program. All these commands and options are explained in more detail in the `README` files that are included in almost all program packages.

options

installation

The configuration of the X Window System software proves even easier. Instead of a makefile, X Window System programs normally contain an `Imakefile`, which is a system-independent makefile. The command `xmkmf` automatically generates a makefile from the `Imakefile`; the makefile contains all important paths and options of the respective system. For such programs it usually suffices to enter the commands `xmkmf`, `make`, and `make install`.

`Imakefile`

`xmkmf`

Configuration

Newer FSF program packages often contain a script named `configure`. This script is invoked first, independently recognizes the operating system, and searches the system for its C compiler, other tools, and libraries. The information found is used during compilation, so that no further adaptation is necessary.

FSF

`configure`

Such `configure` scripts are usually created with the GNU `autoconf` package, which uses the `m4` macro processor to create

`autoconf`

the shell script `configure` from a yet rather readable definition in the file `configure.in`.

Manual adaptation

error If the compiler terminates the make procedure with an error message, then the user must make some manual adaptations. This occurs frequently with programs that do not use the `configure` script. With a little practice, such adaptations are easy to carry out. The following is a potential error message:

```
linux2:/home/tul/tmp> gcc bsp.c
bsp.c: In function 'main':
bsp.c:3: 'errno' undeclared (first use this function)
bsp.c:3: (Each undeclared identifier is reported only once
bsp.c:3: for each function it appears in.)
linux2:/home/tul/tmp>
```

unknown symbol Here the compiler did not recognize the definition of a symbol. This can have several causes. If the symbol is a constant or a macro, then either it is not available under Linux or the header file containing the declaration was not included with `#include`. The same applies to C functions whose prototypes are in header files. The user should first gain an overview of this function and its parameters from the on-line reference Manual page.

search in header file To determine whether a function or another symbol exists under Linux, the system include files, i.e., the C header files of the Linux C library, can be searched. These header files are located in the directory `/usr/include` and can be searched with the following command:

```
find /usr/include -name "*.h" -exec grep "symbol" {} \;
-print
```

`/usr/include` The `find` command seeks all header files in the directory `/usr/include` and all its subdirectories; then `find` invokes the `grep` command, which searches each found file for the symbol. If the undefined symbol is a function that differs on many UNIX systems, then the source code frequently contains alternatives that can be activated with a compiler option such as `-DSYSV` or `-DBSD`. In the source code this could take the following form:

System V and BSD


```
#ifdef SYSV
    function_a (x, y, z);
#else
    function_b (x, y, z);
#endif
```

If the user needs to make modifications in the source code, they should be embedded in `#ifdef` or `#ifndef` `linux` commands. Here `linux` is a symbol that is automatically declared by the compiler when it is invoked under Linux.

```
#ifdef linux
    changes
#else
    old code
#endif
```

If the compiler does not report error messages, but the linker displays error messages on undefined symbols, then the program `nm` provides a valuable aid. It outputs a list of symbols declared and used in object files and libraries. In combination with using `grep` for searching the list of undefined symbols, this makes it possible to determine where a symbol is used and which library contains it:

```
linux1:/home/tul>nm prog1.o | grep "symbol"
```

With a little practice, the user will be able to adapt smaller programs in only a few minutes so that they can be compiled under Linux.

Archiving

After modifications have been made on the source code of a program, the modifications should be stored in a file with the help of the `diff` (difference) command. Then the modifications can be applied at any time to the original program with the `patch` command, and these modifications also can be made available to other users.

The following example shows how to use `diff`:

```
hermes:/usr/src# diff -Nrc tcsh-6.05 tcsh-6.05.patched > tcsh.patch
hermes:/usr/src#
```

To apply the changes in the patch/diff file to the unchanged source code, invoke patch as follows:

```
hermes:/usr/src# patch < tcsh.patch
Hmm... Looks like a new-style context diff to me...
The text leading up to this was:
.....
|diff -Nrc tcsh-6.05/Makefile tcsh-6.05.patched/Makefile
|*** tcsh-6.05/Makefile Thu Jan  1 01:00:00 1970
|--- tcsh-6.05.patched/Makefile Thu Dec 15 13:10:57 1994
|
(Creating file tcsh-6.05/Makefile...)
Patching file tcsh-6.05/Makefile using Plan A...
Hunk #1 succeeded at 1.
Hmm... The next patch looks like a new-style context diff to me...
The text leading up to this was:
.....
|diff -Nrc tcsh-6.05/tc.func.c tcsh-6.05.patched/tc.func.c
|*** tcsh-6.05/tc.func.c      Sun Jun 26 00:02:54 1994
|--- tcsh-6.05.patched/tc.func.c      Wed Dec 14 00:03:25 1994
|
Patching file tcsh-6.05/tc.func.c using Plan A...
Hunk #1 succeeded at 1858.
Hunk #2 succeeded at 1872.
Hmm... The next patch looks like a new-style context diff to me...
The text leading up to this was:
.....
|diff -Nrc tcsh-6.05/tc.who.c tcsh-6.05.patched/tc.who.c
|*** tcsh-6.05/tc.who.c Sun Jun 26 00:02:55 1994
|--- tcsh-6.05.patched/tc.who.c Thu Dec 15 13:10:47 1994
|
Patching file tcsh-6.05/tc.who.c using Plan A...
Hunk #1 succeeded at 250.
Hunk #2 succeeded at 589.
done
```

Reference

apropos terms

Searches the command descriptions in the Manual pages for the terms passed as parameters and displays the descriptions of the appropriate commands. This is equivalent to the command `man -k`. (Also see `whatis`.)

apropos

ar [-]operation [arguments] [position_name]
archive [files]

Processes an archive file, which is usually a C compiler library. With this command you can combine any binary files to a library or extract files from a library. Only one operation may be specified, but multiple arguments are permitted.

ar

Available operations

- d** deletes the files from the archive
- m** moves the files to the archive (with the position depending on additional arguments)
- p** lists the files in the archive
- q** appends the specified files to the end of the archive
- r** replaces the specified files in the archive with the new files
- t** lists the contents of an archive (with argument `v` providing a verbose list output)
- x** extracts all files or only those specified from the archive

Available arguments

- a** places the files in the archive after position name (can be specified with `r` or `m`)
- b** places the files in the archive before position name (can be specified with `r` or `m`)
- c** creates the archive
- i** see argument `b`
- o** retains the original file date on extraction
- s** creates the file table of the archive anew
- u** replaces only files that have been modified (can be specified with `r`)
- v** outputs detailed messages for every operation

at [options] time

at

Executes commands at a certain time. The commands are entered at the standard input device and terminated with EOF (`<Ctrl-d>`). Option `f` permits alternative input from a shell script. A Bourne shell (`/bin/sh`) is used for execution.

Via the option `q` individual jobs can be assigned to different queues (a-z, A-Z), where letters later in the alphabet reflect decreasing priority.

The time can be specified in numeric form (HHMM, HH:MM) or with a keyword such as `noon`, `teatime` (16:00) or `midnight`. Alternatively, the time can be specified as a difference such as `now +3 hours`. Minutes, hours, days, weeks, months, and years are permissible units. If the job is to run on a certain day, then the month (Jan, Feb, Mar, ...) and the year (95, 96, ...) are specified additionally.

Available options

- b** equivalent to the command `batch`
- d** removes the specified jobs from the queue (`atrm`)
- f file** executes the commands in *file*
- l** lists the current user's jobs (`atq`)

-
- m** sends an e-mail to the user when the commands have been completed
 - q *Q*** assigns a job to a particular queue specified in *Q* (a-z, A-Z)
 - V** returns the version number
-

atq [options]

Displays the jobs that are yet to be executed by the user's `at` commands.

atq

Available options

- q *Q*** restricts output to the contents of a specific queue *Q*
 - V** displays the version number of the command
 - v** displays a list of jobs that have been executed, but not deleted
-

atrm [options] jobs

Removes the specified `at` jobs. A job is identified by its job ID, displayed by the `at` or `atq` command.

atrm

Available options

- V** displays the version number
-

awk [options] [program] [-v Var=value ...] [files]

`awk` is a simple interpreter with the combined functionality of `grep` and `sed`. It contains its own C-like language. `awk` is particularly suitable for evaluating ASCII files and for creating scripts for system administration.

awk

Available options

- f *file*** reads the `program` from the specified `file` instead of the command line
- F *c*** sets the delimiting character for fields to `c`
- v *var=value*** assigns the variable `var` the specified `value`

basename**basename** pathname [suffix]

Clears the `path` and optionally a specified file extension and outputs the remaining file names to the standard output device. It is usually used in shell scripts.

bash**bash** [options] [arguments]

A command interpreter similar to Bourne shell and Korn shell.

batch**batch** [options] [time]

Behaves like the `at` command. However, it only executes the specified commands when the system load is low. (Also see `at`.)

Available options

- f file** executes the commands specified in *file*
- m** sends an e-mail to the user when the commands have been completed
- q Q** assigns the job to the queue specified in *Q* (a-z, A-Z)
- V** displays the version number

bc**bc** [options] [files]

Interactive program for computation numbers to another base. `bc` has its own language, which supports the definition of new functions, for example.

Available options

- l** makes the functions of the mathematics library available
- s** causes POSIX-compatible behavior
- w** displays warnings that conform to POSIX

Example

```
zeus:/home/uhl> bc
bc 1.02 (Mar3, 92) Copyright © 1991, 1992 Free Software Foundation, Inc.
This is free software with ABSOLUTELY NO WARRANTY.
For details type 'warranty'.
a=5
b=3
```

```
a*b
15
quit
zeus:/home/uhl>
```

cal [options] [[month] year]

Displays a calendar for the current month or a specified month or year. The number of the year must be given in long form (e.g., 1995) and the month as a number (1-12).

cal

Available options

- j** displays a Julian calendar (with days numbered sequentially)
- y** displays a calendar for the current year

cat [options] [files]

Reads multiple files and outputs them to the standard output device. If no files are specified, the standard input device is read. For this command, output is frequently redirected with >.

cat

Available options

- b** sequentially numbers all nonempty lines
- e** can be specified along with **-v** and outputs “\$” for end of line (EOL)
- n** sequentially numbers all lines
- s** replaces a group of blank lines with a single one
- u** unbuffered output
- v** also outputs control characters and other nonprintable characters
- t** can be specified along with **-v** and outputs “^I” instead of tabs and “^L” instead of page feeds

Example

```
zeus:/home/uhl> cat >file.txt
This is the contents of the file!
<Ctrl-d>
zeus:/home/uhl> cat file.txt
This is the content of the file!
zeus:/home/uhl>
```

cc [options] files

cc

C compiler (see `gcc`).

cd [directory]

cd

Changes the **current directory**. This command is usually included in the shell. If no **directory** is specified, the current user's home directory is assumed by default. Specifying `"-"` returns to the previous directory.

chgrp [options] group files

chgrp

Changes the **group membership of files**. This command can be used by the system administrator or by the owner of the specified **files**. The **group** can be specified in the form of a numeric group ID or as the name of a group.

Available options

- c** displays the names of the **files** whose **group membership** actually changed
 - f** suppresses error messages
 - R** changes the **group membership of files** in subdirectories (recursively)
 - v** describes each change in a verbose manner
-

chmod [options] permissions files

chmod

Changes the **permissions of the specified files**. This command can be used by the system administrator or the owner of the specified **files**. The **permissions** can be specified numerically (octal format) or with a command string. The command string can consist of: the designation for owner (u), group (g), or other (o); the command to add (+), remove (-), or set (=); the permissions to read (r), write (w), or execute (x); and the commands to set or reset the special flags *set user ID* (s) and *sticky* (t).

Available options

- c** displays the names of files whose permissions actually changed
- f** suppresses error messages
- R** also changes the permissions of files in subdirectories (recursively)
- v** describes each change in a verbose manner

Examples

```
chmod u+x file
    adds execution permission for the owner of the file
chmod go-wx files
    removes read and execution permissions for the specified
    files for the group and other users
chmod g+s file
    sets the set group ID flag of the specified file
chmod=r file
    sets the file's permissions to read-only for everyone
chmod 644 file
    allows read and write permissions for the owner and read
    permissions for all others
```

chown [options] owner[:|.group] files

Changes the `owner` and optionally the `group` as well for the specified files. The `owner` and the `group` can be specified as numeric IDs or as names.

chown

Available options

- c** displays the names of the files whose `owner` has actually changed
- f** suppresses error messages
- R** also recursively changes `owners` of files in subdirectories
- v** describes each change in a verbose manner

Reference

cksum

cksum [files]

Computes CRC checksums for the specified *files* and displays these along with their file size and file name.

clear

clear

Clears the screen.

cmp

cmp [options] file1 [file2]

Compares the contents of two files bitwise. If the files are identical, 0 is returned; otherwise, 1. If "-" is specified as the file name, then the command reads from the standard input device. The same applies if *file2* is not specified.

Available options

- c** displays the characters that are different
 - l** displays the offset and octal values of deviating bytes
 - s** suppresses all screen output
-

comm

comm [options] file1 file2

Compares two linewise presorted files. Without additional options the output is in three columns: the first column contains the lines that occur only in *file1*, the second column displays all lines that occur exclusively in *file2*, and the third column contains all common lines.

Available options

- 1** suppresses column 1
- 2** suppresses column 2
- 3** suppresses column 3

compress [options] [files]

Compresses the specified files using the Lempel-Ziv method. The compression is indicated by appending “.z” to the file name. All other file attributes are retained.

compress

Available options

- b n** restricts to *n* the number of bits that may be used for coding
 - c** outputs the results to the standard output device and does not change any files
 - f** compresses without confirmation if the target file already exists
 - r** also compresses files in subdirectories (recursively)
 - v** provides a verbose status report
 - V** displays the version number of the program
-

cp [options] file1 file2

cp [options] files directory

Copies file1 to file2 or the specified files into the directory. If the target file (file2) already exists, it is overwritten (although option **-i** requires confirmation).

cp

Available options

- a** combination of **-d**, **-p** and **-r**
- b** creates a backup of files before overwriting them
- d** maintains symbolic and hard links during copying
- f** forces a copy and overwrites existing files
- i** asks for confirmation before overwriting an existing file
- l** creates a hard link rather than a copy of a file
- P** copies files into a target directory hierarchy (which is created if necessary)
- p** also copies the permissions and modification times of the files
- r** recursively copies subdirectories and their contents
- R** see **-r**

- s** creates a symbolic link rather than a copy of a *file*
- S *suffix*** changes the extension for backup files to *suffix*
- u** prevents the overwriting of a file that has the same name and a newer date
- v** displays the name of each *file* on copying
- x** ignores directories on any file system different from the source file
- V {numbered, existing, simple}** determines the kind of version control:
 - numbered** always creates a numbered backup
 - existing** creates a numbered backup only for files for which such a backup already exists, and in all other cases creates a simple backup
 - simple** always creates a simple backup

cpio *options* [*arguments*]

cpio

Copies files into an archive, displays the contents of an archive, or extracts files from an archive. The archives can be on magnetic tape, hard disk, or floppy disks. *cpio* has three modes of operation, selected by the options **-i** (*copy in* = unpack), **-o** (*copy out* = pack), and **-p** (*copy pass* = copy from directories). *cpio* was designed to work with the *find* command.

Available options

- 0** accepts file names terminated with null instead of newline (copy out and copy pass modes)
- a** resets the access times of files that are read so that the reading cannot be discerned from the file date
- A** adds files to an existing archive (with options **-O** or **-F**)
- b** during extraction (copy in), exchanges words and half-words (little/big endian)
- B** increases the input/output buffer from 512 to 5120 bytes
- c** uses the (old) portable ASCII format for the file headers
- C *n*** sets the input/output buffer to *n* bytes
- d** automatically creates the necessary subdirectories during extraction

- E *file*** extracts the files whose names are in *file* (copy in)
- f** copies only the files that do not match the specified search pattern
- F *file*** uses the specified *file* as an archive instead of the standard output device. *file* can also contain the name of a host in order to write the archive to a remote magnetic tape (-F zeus:/dev/tape)
- H *format*** reads/writes header information in the specified *format*:
 - bin** old binary format
 - odc** old portable format (POSIX.1)
 - newc** new portable format (SVR4)
 - crc** new SVR4 format with CRC checksums
 - tar** old tar-compatible format
 - ustar** POSIX.1-compatible tar format
 - hpbinary** old HP UNIX binary format
 - hpodc** portable HP UNIX format
- i** puts `cpio` in copy-in mode (extraction of an archive)
- I *file*** uses the specified file instead of the standard input device. A host name (zeus:/dev/tape) can be specified, for example, to access an archive on a remote magnetic tape drive
- L** dereferences symbolic links, meaning that not the link but the file to which the link refers is copied
- m** the original modification date of a file is retained on creation of a new file
- M *msg*** enables multivolume archives. If a storage medium is full, the message `msg` is displayed on the screen. The variable `%d` can be used within the message to display the current number of the medium
- n** on display of the directory listing, the UID and GID are displayed as numeric values
- o** puts `cpio` in copy-out mode (creation of an archive)
- O *file*** uses the specified file instead of the standard output device. A machine name (zeus:/dev/tape) can be specified, for example, to access an archive on a remote magnetic tape drive
- p** puts `cpio` in copy-pass mode (copy directories locally)

- r [user][:][group]** changes the file owner in copy-out and copy-pass modes and can only be used by the administrator
- s** exchange bytes in copy-in mode
- S** exchange halfwords in copy-in mode
- t list** displays a *list* of the contents of an archive
- u** permits overwriting of files with the same name and an older version
- v** displays a list of file names. A verbose version can be obtained by combining with the option **-t**.
- V** displays a period (".") for each processed file

Examples

```
find. -name '*.txt' -print | cpio -ocv > /dev/tape
```

backs up all files that end with "txt" into an archive on magnetic tape

```
cpio -icdv < /dev/tape
```

extracts all files from magnetic tape to the hard disk

```
find. -print | cpio -pdv /tmp
```

copies all files from the current directory to /tmp

```
crontab [-u user] file
```

```
crontab [-u user] operations
```

crontab

Replaces, edits, lists, or deletes a user's `crontab` file. The administrator can process any user's `crontab` file by using the option `-u`.

Available operations

- e** edits the `crontab` file in the default editor (environment variable `EDITOR`)
- l** lists a user's `crontab` file
- r** deletes the `crontab` file

```
csh [options] [arguments]
```

cs

Command interpreter with a syntax based on C.

csplit [options] file [expression]

Splits the specified file into multiple smaller files and displays the sizes of the generated files. If the specified file name is "-", then data are read from the standard input device. The locations for the splitting can be specified by an optional expression of the following form:

number specifies the number of lines after which a new output file is to be created

/regex[offset] regular expression that specifies the splitting locations; an optional positive (+) or negative (-) line offset can be defined

%regex %[offset] like the above expression, but in this case the specified section is skipped rather than written to a file

{repetitions} induces the repeated application of an expression to which it is appended. If an asterisk (*) is specified instead of a number, then the expression is applied until the end of the input file is reached

Available options

- f prefix** specifies the prefix for the generated output files
- b suffix** changes the suffix of the generated files. The format of suffix is based on the format commands of `printf`. `%d` sets the number of the output file in decimal form, while `%x` results in hexadecimal representation
- k** already generated files are preserved, even if the command is aborted
- n n** length of the sequential number in the name of the output files (default 2)
- q** suppresses screen output
- s** see `-q`
- z** suppresses the generation of files of length 0

Examples

```
csplit -k linux.txt '%cut%' {30}
```

splits the file `linux.txt` at positions "cut" into at most 30 output files

```
csplit -k list.txt 10 {100}
```

splits the file `list.txt` into at most 100 files with 10 lines each

ctags

ctags [options] files

Reads the specified C, Fortran, Pascal, LaTeX, or Lisp source files and generates a list of functions and macros defined therein. This list can be processed in the `vi` or `emacs` editor. A keyword list (tag file) is generated with the name `tags` in the current directory.

Available options

- a** appends the names found to an existing list
- B** generates a search pattern for a backwards search in `vi`
- C** activates C++ mode, where `.c` and `.h` files are treated as C++ code
- d** generates entries for preprocessor definitions as well
- f file** writes the names it finds to `file`. If `-f` is not specified, then the file `tags` is used
- F** generates a search pattern for forward search in `vi` (default)
- H** displays a help text
- i file** continues the search for a tag in the specified `file`
- o file** changes the name of the output `file`
- S** ignores indents
- t** also generates a tag for type definitions
- T** also generates a tag for type definitions, structures, enumerations, and C++ member functions
- u** the tag list is updated
- v** generates an index file in `vgrind` format and outputs it to the standard output device
- V** displays the version number
- w** suppresses warnings about duplicate entries
- x** generates a cross reference list in `cxref` format and outputs it to the standard output device

cut options [files]

Cuts a series of fields or columns from a line of the input file. One of the options **-b**, **-c** or **-f** must be specified. Each of these options expects a list that can contain numbers separated by commas or fields defined by hyphens.

cut

Available options

- b list** selects the character at the position defined in *list*
- c list** selects the columns specified in *list*
- d c** is output together with **-f** to specify the field delimiter character (*c*)
- f list** selects the fields (separated by tabulators or the delimiting character) from *list*
- s** restricts output to lines that contain the field separator

Example

```
cut -d: -f1, 3 /etc/passwd
    outputs the login names and user IDs of all users
```

date [options] [+format]

date [options] [string]

In the former form, the current date and time are returned in a *format* that can be provided optionally. With the second form, the system administrator can set the system time.

date

Output format

- % %** percent sign
- %n** new line
- %t** tabulator
- %H** hour (00 .. 23)
- %I** hour (01-12)
- %k** hour (0 .. 23)
- %l** hour (1 .. 12)
- %M** minute (00 .. 59)

%p	AM or PM
%r	time in 12-hour format (hh:mm:ss[AM PM])
%s	seconds since January 1, 1970, 0:00
%S	seconds (00 .. 59)
%T	time in 24-hour format
%X	time in local format
%Z	time zone, if defined, else empty
%a	local abbreviation of day name
%A	local name of day of week
%b	local abbreviation of month name (Jan ... Dec)
%B	local month name (January ... December)
%c	local date with time and time zone
%d	day of month (01 .. 31)
%D	date (mm/dd/yy)
%h	identical to %b
%j	sequential day of the year (001 .. 366)
%m	month as number (01 .. 12)
%U	week as number (0053) where Sunday is the first day
%w	day of week as number (0 .. 6)
%W	week as number (00 .. 53) where Monday is the first day
%x	local representation of the date (dd/mm/yy)
%y	last two digits of the year (00 .. 99)
%Y	year (1995 ...)

Format of the string to set the time

DD	day of month
hh	hour
mm	minute
CC	first two digits of year (=century)
YY	last two digits of year
ss	seconds

Available options

- d date** outputs the specified date (which can contain the month name, time zone, ... etc.)
- s date** sets the date in arbitrary format (which can contain the month name, time zone, ... etc.)

-u ignores time zone and uses UTC (Universal Coordinated Time)

dd [options=value ...]

copies from the standard input device or a specified file to the standard output device or another specified file. The most frequent options are **if** to specify the input file and **of** to specify the output file. **dd**, for example, can be used to write a kernel image file directly onto a diskette or to make a boot diskette from a disk image.

dd

Available options

bs=n sets the block size for input and output to *n* bytes. Optionally, *n* can be specified with units, e.g., 8k for 8 kilobytes

cbs=n determines the size of a field in converting to bytes

conv=flags converts the input according to the following arguments:

ascii EBCDIC to ASCII conversion

ebcdic ASCII to EBCDIC conversion

ibm ASCII to IBM EBCDIC conversion

block converts variable-length fields to fields of length *cbs* and fills the spaces with blanks

unblock converts fixed-length fields (*cbs*) to variable-length fields

lcase converts uppercase to lowercase letters

ucase converts lowercase to uppercase letters

swap swaps every two bytes of the input file

noerror ignores errors during reading

notrunc does not truncate the output file

sync fills spaces in the input blocks of size *ibs* with zeros

count=n copies only *n* blocks

if=file specifies the input *file*

of=file specifies the output *file*

ibs=n sets the size of the input buffer

obs=n sets the size of the output buffer

skip=n skips *n* blocks of input

df

```
df [options] [paths]
```

Outputs the number of occupied and free blocks of file system. If no `path` is specified, then a list of all current file systems is output. If a `path` is specified, an overview is provided of the associated file systems. Alternatively, the direct `path` of a device (`/dev/hda1`) on which the file system is located can be specified. Normally only real file systems with a storage volume greater than zero are output.

Available options

- a** displays all current file systems, including those of size zero
 - i** instead of block information, displays i-node statistics
 - k** uses a block size of one kilobyte (default)
 - P** uses POSIX output format
 - t *type*** restricts output to file systems of a certain type
 - x *Type*** ignores file systems of a certain type during output
-

diff

```
diff [options] file1 file2
```

Compares two files or all files in two directories. If one of the two paths is specified as `"-"`, then the files are expected from the standard input device. The output of `diff` lists all lines that occur in only one file or that are different. This output can be used by `patch` to make changes in files. Another alternative for comparing or merging files is the Emacs Lisp program `ediff`.

Available options

- a** treats all input files as text files and compares linewise
- b** ignores differences in the number of blanks (at the end of a line as well)
- B** ignores blanks
- c** generates output with three lines of context around each difference
- C *n*** like `-c`, but `n` lines of context are output around each difference
- d** uses a better, although slower, algorithm for file comparison

-
- D *name*** mixes the two files and inserts appropriate preprocessor instructions (`#ifdef name`) to make the two versions distinguishable. If *name* is defined during compilation, then the version is output to `file1`, otherwise to `file2`
 - e** outputs instructions for the `ed` editor to be able to generate `file2` from `file1`
 - f** like option `-e`, but reversed, yet it cannot be used as an `ed` script
 - h** is ignored
 - H** uses heuristics to increase the speed
 - i** ignores differences in upper/lower case
 - l** (only when comparing whole directories) the output can be processed with the command `pr` so that each file begins on a new page
 - n** generates output in RCS format
 - N** in comparing two directories, missing files are considered as existing, but empty
 - q** simply reports whether the files are different
 - r** (only when comparing whole directories) subdirectories are handled recursively and all files are compared
 - s** reports whether two files are identical
 - S *file*** starts comparing directories with a certain file
 - t** replaces tabs with blanks
 - T** outputs a tab instead of a blank at the start of every output line
 - u** generates output in GNU-specific “unified” format
 - v** displays the version number
 - w** ignores blanks and tabs when comparing lines
 - x *pattern*** ignores files and subdirectories that match the specified *pattern* (when comparing whole directories)
 - y** outputs in easy-to-read, two-column format
-

diff3 [*options*] *file1 file2 file3*

Compares three files linewise.

Available options

- a** linewise comparison treating all input files as text files
- A** inserts all changes between `file2` and `file3` in `file1` and marks conflicts
- e** generates a script for the `ed` editor that integrate all changes from `file2` to `file3` in `file1`
- E** like option `-e`, but the output is less verbose
- i** generates `w` and `q` commands at the end of a generated `ed` script
- m** applies the edit script to `file1` and displays it
- T** outputs a tab instead of a blank at the start of every output line
- v** outputs the version number of the command
- x** like option `-e`, but only overlapping changes are output
- X** like option `-E`, but only overlapping changes are output
- 3** like option `-e`, but only nonoverlapping changes are output

dirname `pathname`

dirname

Extracts the directory part of a complete path specification (counterpart to `basename`). If the path does not contain a file at the end, then `."` is returned.

du [`options`] [`files` | `directories`]

du

Outputs the sizes of the specified files or directories.

Available options

- a** outputs the sizes of all files, not just directories
- b** outputs the file size in bytes
- k** outputs the file size in kilobytes
- l** outputs the sizes of (hard) linked files, even if this means handling them doubly
- s** outputs the total size of all files and subdirectories
- x** ignores directories in different file systems

```
echo [-n] [text]
```

This command is usually built into shells. It outputs `text` to the standard output device. The option `-n` suppresses the output of the newline character.

echo

```
ed [options] [file]
```

This is an antiquated standard editor which, apart from its use with the `diff` command, is no longer of any importance.

ed

```
egrep
```

See grep.

egrep

```
env [options] [variable=value] [command]
```

If invoked without parameters, this command produces a list of all environment variables. In addition, this command permits starting commands in a modified environment. In the command line, new variables can be defined or existing ones removed.

env

Available options

- i** ignores the inherited environment
 - u name** removes the specified environment variable
-

```
expr arg1 operator arg2 [operator arg3 ... ]
```

Evaluates an expression and outputs the result to the standard output device. Expressions can be numeric, logical, or relational. This command is usually used in shell scripts.

expr

Arithmetic operators

`+`, `-`, `*`, `/`, `%` (modular rest)

Relational operators

`=`, `!=`, `>`, `>=`, `<`, `<=`

Logical operators

| (or), & (and),
: (seek Arg2 as a regular expression in Arg1)

Examples

```
expr 7+8/2
```

evaluates to 7 (integer arithmetic, left to right!)

```
expr $s="hello"
```

evaluates to 1 if *s* contains the string "hello," else 0

false

false

This command does nothing and returns *false* (not 0). (Also see *true*.)

fdformat [-n] device

fdformat

Executes low-level formatting of a diskette. The required parameter is the path of the corresponding *device*. The first drive is addressed as */dev/fd0XXX*; the second, as */dev/fd1XXX*. The *-n* option suppresses subsequent verification of the diskette.

Device	Sectors	Tracks	Size	Capacity (KB)
<i>/dev/fd0h1200</i>	15	80	5 1/4	1200
<i>/dev/fd0D720</i>	9	80	3 1/2	720
<i>/dev/fd0H1440</i>	18	80	3 1/2	1440
<i>/dev/fd0E2880</i>	36	80	3 1/2	2880

fgrep

fgrep

See *grep*

file [options] files

file

Outputs the types of specified *files*. The file type is recognized on the basis of an extensible rule file (*/etc/magic*).

Available options

- c** for checking the rule file
- f *file*** examines the files listed in *file*
- m *file*** uses the specified rule *file* instead of `/etc/magic`
- L** also follows symbolic links
- z** enables the processing of compressed files

find `pathname constraints`

This command recursively searches in directories for files that meet all the specified `constraints`. The list of `constraints` is evaluated from left to right. Individual `constraints` can be negated by a preceding exclamation mark (!). An OR conjunction between two expression is defined with `-o`. `find` is particularly useful in combination with other commands (e.g., `cpio`).

find

Numeric specifications can be represented in three ways

- +n** value larger than `n`
- n** value equal to `n`
- n** value smaller than `n`

Available options (always true)

- depth** files contained in a directory are processed before the directory itself
- follow** also branches to directories indicated by symbolic links (follows symbolic links)

Possible constraints

- amin *n*** files accessed in the last *n* minutes
- anewer *file*** files accessed more recently than the specified *file*
- atime *n*** files that were last accessed *n* days ago
- ctime *n*** files that were last modified (either the file itself, the permissions or the owner) *n* days ago
- fstype *type*** files in a certain *type* of file system (e.g., `ext2`, `msdos`, `proc`)
- group *group*** files belonging to a certain *group* (name or ID)

- inum *n*** files with i-node number *n*
- links *n*** files that possess *n* links
- local** files physically stored on the local system
- mtime *n*** files that were last modified (only the file itself) *n* days ago
- name *pattern*** the names of the files match the specified wildcard *pattern*
- newer *file*** the last modification of the files must be more recent than the specified *file* (see also `mtime`)
- nogroup** files whose group does not exist in `/etc/groups`
- nouser** files whose owner does not exist in `/etc/passwd`
- perm *nnn*** the permissions of the files must match the octal representation *nnn*
- size *n* [*c*, *k*]** files of size *n* blocks, *n* bytes, or *n* kilobytes
- type *c*** files of type *c*, where *c* must derive from the following list:
 - b** block special file
 - c** character special file
 - d** directory
 - p** FIFO or named pipe
 - l** symbolic link
 - f** normal file
- user *user*** files belonging to a certain *user* (name or ID)

Possible actions

- exec *command* {};** executes the *command* for each file and tests whether the return code is 0. During execution, {} is replaced by the name of the current file
- ok *command* {};** like `exec`, but the user must confirm the *commands* with "y"
- print** outputs files or directories found
- printf *format*** like `-print`, but the format of the output can be influenced by a *format* string

Examples

```
find. -type f -print
```

outputs all normal files in the current directory and its subdirectories

```
find ./usr/include-typef" -execgrep "read" {} \;-
print
```

searches all normal files in /usr/include for the character string "read"

finger [options] [user]

Provides information on users. *user* can be in the form of *name*, *name@host* or *@host*. In the first two forms the names of the users, the times, the last logins, and additional information are output. If the file *.plan* or *.project* exists in the *user*'s home directory, then it is also displayed. If only one host is specified with *@host*, then all users are listed who are currently logged into that system.

finger

Available options

- l** forces verbose output (at *@host*)
 - m** the specified *user* must exactly match the user name. Without this option the specified name is also compared with the full name of the user as stored in the file */etc/passwd*
 - p** the files *.plan* and *.project* of the respective user are not displayed
 - s** forces brief output format
-

ftp [options] [*host*]

This program is for transferring files with the ftp protocol. The *host* can be specified by name or IP address. If no *host* is specified, then the program responds with a prompt that allows entry of ftp commands. Entering *help* evokes a help text.

ftp

Available options

- d** debug mode
- g** toggles off the use of wildcards for file names
- i** turns off queries (*mget*, *mput*)
- n** suppresses automatic login on computer listed in the file *.netrc*

-v displays all the ftp server's messages

gcc

gcc [options] [files]

In addition to C, C++, and Objective C, the GNU C compiler supports back ends for other languages such as Ada and Pascal. A more detailed description of these features can be found in the GNU Info documents.

grep

grep [options] regexp [files]

Searches *files* or data from the standard input device linewise for a regular expression (*regexp*) and outputs the found lines to the standard output device.

Available options

- b** additionally outputs the byte position where the expression was found
 - c** outputs only the number of lines in which the expression was found
 - h** suppresses the output of file names
 - i** ignores differences in size and (upper/lower) case
 - l** outputs only the names, but not the lines, of files in which the expression was found
 - n** outputs the line numbers of found lines
 - s** suppresses error messages in case a file does not exist or cannot be opened
 - v** searches for lines that do not contain the regular expression
-

groff

groff [options] [files]

groff is the GNU variant of *nroff* and *troff*. The command serves to format Manual pages and other documents that are available in the appropriate format. Additional preprocessors such as *eqn* or *tbl* are integrated in *groff* and can be activated with options. The results can be stored in ASCII, DVI or PostScript format. In formatting documents, it is important to specify the macro package used. For

Manual pages, for example, the option `-man` would be specified. The formatted file is written to the standard output device.

Available options

- a** outputs pure ASCII format
- e** activates the `eqn` preprocessor
- E** suppresses error messages
- h** outputs help text
- m *macro*** uses a special *macro* package for formatting
 - `-man` macros for Manual pages
 - `-ms` `ms` macro package
- p** activates `pic` preprocessor
- s** activates `soelim` preprocessor
- t** activates `tab` preprocessor
- T *format*** specifies output *format* (`ascii`, `ps`, `dvi`) and
- v** outputs the version number

Example

```
zeus:/home/uhl> groff -man -Tps ls.1 > ls.ps
```

formats the Manual page `ls.1` with the corresponding macro package and outputs the result in PostScript format to the file `ls.ps`

groups [*user*]

Outputs the groups to which the specified *user* belongs. A parameterless invocation lists all the current user's own groups, and with a parameter the groups of the specified *user*. The command evaluates the files `/etc/passwd` and `/etc/groups`.

groups

gzip [*options*] [*files*]

Compresses or decompresses *files* using the LZ77 method and adds the extension `.gz` to the file name. If no file (or `-`) is specified, then the standard input device is read and output goes to the standard output device. `gzip` can also decompress files packed with `compress` (ending in `.Z`).

gzip

Available options

- a** adapts the end of line in ASCII texts to the respective system (CRLF or LF)
- c** writes the results to the standard output device without overwriting the input file
- d** decompresses packed files
- f** forces an overwrite of existing files
- l** for a compressed file, displays its size in packed and unpacked form, the compression rate, and the name of the original file
- q** suppresses warnings
- r** recursively works through subdirectories
- S .suffix** changes the file *suffix* for compressed files
- v** displays the name and the compression rate for each file
- #** sets the quality of compression from 1 (poor) to 9 (good), where default compression is 6

head [options] [files]

head

Displays the first 10 lines of the specified (text) *files*. On specification of multiple files, the file name precedes the file contents in the output.

Available options

- #** changes the number of lines to be output to the specified value
- c n[b|k|m]** outputs the first *n* bytes, where the specification can be made in bytes (*b*), kilobytes (*k*), or megabytes (*m*)
- q** suppresses the output of file name

hostname [name]

hostname

Parameterless invocation displays the *name* of the host; otherwise the specified *name* is set. The host name is usually set on system startup and requires superuser permissions.

id [options]

Displays the real and effective user ID (UID) and all groups (GID) of the current user.

id

Available options

- g** displays only GID
 - G** displays only the additional groups to which a user belongs
 - n** displays GID or UID as name (only in combination with options **-g**, **-u**, **-G**)
 - r** displays the real instead of the effective GID (only in combination with options **-g**, **-u**, **-G**)
 - u** displays only UID
-

join [options] file1 file2

Joins two alphabetically sorted ASCII files via a key. Lines with identical keys are joined and written to the standard output device. Keys must be separated by blanks or tabs. If no further options are specified, the first column is used as the key.

join

Available options

- a [*n*]** adds an empty line to the output if one line of file *n* (1 or 2) does not have a matching key in the other file
 - e *string*** replaces empty output fields with the specified character *string*
 - j *n m*** uses column *m* of file *n* (1 or 2) as the key
 - o *n.m*** displays only column *m* of file *n*
 - t *z*** uses character *z* as field delimiter (input/output)
-

kill [options] processes

This command is usually built into a shell. It sends a signal to one or more processes. Without further options, a `TERM` signal is sent, which orders a process to terminate. Only the system administrator can send signals to processes that she/he does not own. The processes

kill

are specified with their process number (PID). The signals can be specified numerically or symbolically.

Available options

- l** lists all signal names
- signal** sends a certain *signal* to the specified *processes*. The following signals are useful in this context:

No.	Name	Explanation
1	SIGHUP	Generated on interruption of a terminal connection. For many daemons it serves to read the configuration files anew
2	SIGINT	Equivalent to entering <Ctrl-c>
3	SIGQUIT	Terminates a process and invokes a core dump
9	SIGKILL	Terminates a process. This signal cannot be intercepted
15	SIGTERM	Terminates a process (default).
10	SIGUSR1	User-specific signal whose meaning is different in each application
12	SIGUSR2	See SIGUSR1

ksh [options] [arguments]

ksh

See bash.

last [options] [attribute]

last

Provides information from the login statistics (/etc/wtmp). Without additional arguments, it outputs a list of all login, logout, shutdown, and reboot activities. This list contains the name of the user or the event, the login terminal, the login host, and the time. A selection can be limited to certain entries by specifying search attributes (name, login terminal).

Available options

- #** limits output to a certain number of lines
- f *file*** uses the specified *file* instead of `/etc/wtmp` as the data base
- t *terminal*** lists only logins entered from a particular *terminal*
- h *computer*** lists only logins entered from a certain *computer*

Example

```
hermes:/root# last uhl
uhl  tty4    mobby   Sun Jan 29 17:24   still logged in
uhl  tty2    tonne   Sun Jan 29 16:32 - 16:47 (00:15)

wtmp begins Sun Jan 29 15:18
hermes:/root#
```

ld [options] object_files

The linker links individual object files to an executable program. It is seldom invoked directly. Normally the C compiler or the `make` command automatically invokes the linker.

ld

ldd [options] [programs]

Lists the dynamic libraries that a program needs.

ldd

Available options

- d** carries out a relocation and lists missing functions (only ELF format)
- r** carries out a relocation for data and program code and lists missing objects (only ELF format)
- v** outputs the version number of the command
- V** outputs the version number of the dynamic linker (ld.so)

lex [options] [files]

The scanner generator creates the output file `lex.yy.c` from a scanner grammar as input file.

lex

ln

```
ln [options] path target_path
```

Creates a link. Without options, it creates a hard link to a file. With the option `-s` a symbolic link is created that could also point to a directory. If `target_path` already exists and is a file, then an error message is output. Only with the option `-f` is this file overwritten. If `target_path` is a directory, then the links are created in this directory.

Available options

-f any existing files are overwritten without confirmation
-s symbolic links are created

lpc

```
lpc [command [argument]]
```

Serves to control printer spoolers. It enables the activation and deactivation of individual printers and their printing queues, shifting printer jobs within the printing queues, and outputting status information. Invoking `lpc` without an argument produces an interactive command modus. Alternatively, these commands can also be passed to `lpc` on invocation.

Available commands

help displays a list of available commands
abort {all | printer} terminates active spooler(s) and disables the corresponding *printer(s)*
clean {all | printer} removes all incomplete files from the specified *printer* queue(s)
disable {all | printer} disables the corresponding *printer(s)*
down {all | printer} message turns off the specified queue, disables *printer(s)* and writes the specified *message* in the printer status file. This message is output on invocation of `lpq`.
enable {all | printer} enables the specified printer queue(s) and permits the addition of new jobs
exit, quit ends the `lpc` program

restart {*all* | *printer*} attempts to restart *printer* daemon(s)
start {*all* | *printer*} activates *printer*(s) and starts *printer* daemon(s) for the specified *printer*(s)
status {*all* | *printer*} outputs status information on currently active *printer* daemon(s) and queue(s)
stop {*all* | *printer*} stops the *printer* daemon on completion of the current job and disables the *printer*
topq printer [job#] [user] places the specified job at the head of the queue
up {*all* | *printer*} activates queue(s) and starts *printer* daemon(s)

lpq [options] [job#s] [user]

Provides information on the current status of printer queues.

lpq

Available options

-l verbose status report on each job
-P name selects a printer queue

lpr [options] [files]

Sends *files* to a printer queue. Alternatively, data can be printed via the standard input device. Invocation without options outputs to the queue *lp*.

lpr

Available options

-# n creates *n* copies of the specified documents
-C text prints a job classification on the title page
-h suppresses the output of a header before a print job
-J job prints a *job* name on the title page
-m sends a mail to the user on completion of the job
-P name selects the specified printer queue
-r deletes the file after printing (with option *-s*)
-s file is not spooled but linked. Thus the printer file must not be deleted during printing.
-U user prints the *user* name on the title page

lprm

```
lprm [options] [job#s] [user]
```

Removes entries from a printer queue. *Job numbers* or *user* names can be specified as selection criteria. If no argument is specified, the active job is removed.

Available options

- removes all entries from a queue
 - P name** selects the specified printer queue
-

ls

```
ls [options] [files]
```

Displays the contents of directories or lists specific *files*. If no *files* are specified, the contents of the current directory are listed. If *files* are specified, then only the such files are listed that match the file name (with wildcards).

Available options

- a** displays all *files*, including those beginning with a period
- A** like option **-a**, but suppresses the entries *"."* and *".."*
- B** ignores backup *files* that end in tilde (*~*)
- b** displays nonprintable characters as octal numbers
- c** sorts *files* by time of last status change
- C** displays only file names, but in multiple columns (default)
- d** on specification of a directory name, lists only the directory itself, not its contents
- f** unsorted output
- F** appends a special character to each file name to indicate the file's type (normal file, directory, executable file, link, ...)
- G** suppresses the output of group in long format
- i** displays the associated node for each *file*
- k** displays file size in kilobytes
- l** long format displays every *file* in a line along with its permissions, owner, group, size, etc.
- L** for symbolic links, shows the file or directory to which the link points rather than the link itself

-
- m** lists file names linewise, separated by commas
 - n** lists UID and GID numerically
 - r** lists files sorted backward
 - R** recursively lists subdirectories and their contents
 - s** lists file size in kilobytes before the file name
 - S** sorts list by file size
 - t** sorts list by date of last modification, with newer files coming first
 - u** sorts list by time of last access
 - x** lists files in horizontally sorted columns
 - X** lists files sorted by file extension
-

m4 [options] [files]

This macro processor is used for various program files, in the GNU Autoconf system, and for `fvwm` configuration files. The language is described in the GNU Info system.

m4

mail [options] [addresses]

This program is for reading and sending e-mail. Users should instead use the program `pine` or a graphical mail reader. However, `mail` proves superb for simply sending text files because the contents can be transferred via the standard input device.

mail

Example

```
maillinux@fh-heilbronn.de < critique.txt
```

make [options] [targets]

Reads a makefile and updates one or more `targets`. `make` is usually used for compiling of source files. It is described in detail in the GNU Info system.

make

Available options

- C *directory*** changes to the specified *subdirectory* before a makefile is read
- d** provides additional debugging information
- e** environment variables overwrite corresponding variables in the makefile
- f *makefile*** uses the specified *makefile*
- I *directory*** searches in the specified *directory* for imported makefiles
- k** on error, aborts only the current *target*, not the complete make process
- n** only outputs commands without executing them
- p** outputs internal macro definitions
- r** uses no default rules
- s** suppresses screen output
- t** provides *files* to be processed with the current date without executing the corresponding operation
- w** displays the current working directory before and after executing an operation

man [options] [[section] name]

man

Displays On-line Manual pages pagewise on the screen. These pages are located in a subdirectory under `/usr/man` or in other directories listed in the environment variable `MANPATH`.

Available options

- a** displays all Manual pages that match the specified name
- f** equivalent to the command `whatis`
- h** displays a help page
- k** equivalent to the command `apropos`
- M *path*** specifies a list of additional directories in which to search for Manual pages (see `MANPATH`)
- w** displays not the contents but the access path of a Manual page

mesg [y | n]

Determines whether other users can write messages on the terminal with `write`. If `mesg` is invoked without options, the current status is displayed.

mesg

mkdir [options] directories

Creates directories.

Available options

- m perms** creates a new directory with the specified *permissions*
- p** if a directory path is specified where individual subdirectories do not exist, then these are created also

mkdir

more [options] [files]

Displays *files* by (screen) page. <Enter> scrolls one line down and the space bar advances to the next screen page. <h> displays help with all commands and <q> quits the `more` command. If no *file* is specified, then `more` reads from the standard input device.

more

Available options

- +#** begins with the specified line number
- d** displays the message "Press space to continue, 'q' to quit" at the end of a screen page
- f** counts logical rather than screen lines for page breaks and counts broken lines only once
- l** ignores form-feed control character (^L)
- s** suppresses the output of multiple neighboring blanks
- u** suppresses underlining

mttools

This is a group of commands that permit simple access to MS-DOS file systems. Normally these are used to handle diskettes. Note that

mttools

access to a DOS partition of a hard disk is simpler if it is mounted (see `mount`). The individual commands largely correspond to the DOS commands. This means that floppy disk drives can be accessed with DOS's usual letter designations (A:, B:) if the drives were correctly configured in the file `/etc/mtools`.

Commands

mattrib modifies file attributes
mcd changes the current directory
mcopy copies files
mdel deletes files
mdir displays a directory listing
mformat formats a low-level formatted diskette with a DOS file system
mlabel changes the volume label
mmd creates a subdirectory
mrd removes a subdirectory
mren renames a file
mtype displays the contents of a file

mount

mount [*options*] [*device*] [*mount_location*]

Links new file systems into the directory tree. A file system is attached to the UNIX file tree at a defined `mount location`. Unspecified parameters are taken from the entries of the file `/etc/fstab`.

Available options

-a automatically mounts all file systems specified in `/etc/fstab`
-f suppresses the actual `mount` system invocation (practical with option `-v`)
-n suppresses entries in `/etc/mtab`
-o *opts* additional *options* that depend on the respective file system

General options

async all input and output is asynchronous
auto the file system can be mounted with the `-a` option

defaults standard options: `rw`, `suid`, `dev`, `exec`, `auto`, `nouser`, `async`

dev permits the use of character- and block-oriented devices

exec permits execution of commands

noauto can only be mounted explicitly, but not with the option `-a`,

nodev suppresses the use of character- and block-oriented devices

noexec suppresses the execution of commands

nosuid SUID and SGID bits have no effect

nouser forbids a normal user to mount file systems

remount permits remounting of a file systems, e.g., to change mount options

ro mounts the file system as read-only. This option must be specified to mount CD-ROM file systems.

rw mounts the file system for reading and writing

suid enables the execution of SUID and SGID commands

sync all input and output operations are synchronous

user permits a normal user to mount the file system

File-system-specific options

case={lower | asis} (hpfs) sets (upper/lower) case sensitivity

check=value (ext2) enables the choice of consistency checks before mounting a file systems

none no consistency checks

normal check i-node and block bitmap (default)

strict also checks consistency of free blocks

check=value (msdos) determines the form for specifying file names

relaxed case insensitive, long file names truncated

normal special characters (`*`, `?`, `<`, ...) not accepted (default)

strict no long file names and no special characters

conv=value determines whether end-of-line (EOL) character is converted on access to file system (msdos, hpfs, iso9660)

binary no EOL conversion (default)

text CRLF/LF conversion for all files

auto no conversion on files with the following extensions: `exe`, `com`, `bin`, `app`, `sys`, `drv`, `ovl`, `ovr`, `obj`, `lib`,

dll, pif, arc, zip, lha, zoo, tar, z, arj, tz, taz,
 tzp, tpz, gif, bmp, tif, gl, jpg, pcx, tfm, vf, gf,
 pk, pxl, dvi

- block=*value*** specifies block size for iso9660 file systems
- cruft** sets the *cruft* flag to overcome an error in certain CD-ROM premastering programs (iso9660)
- debug** creates debug messages (ext2, msdos)
- errors=*value*** determines error handling (ext2)
- continue** no special error handling (default)
 - remount ro** file system is remounted as read-only
 - panic** on error, force a kernel panic
- fat=*value*** overwrites the automatically detected *value* for the FAT type (available values being 12 and 16) (msdos)
- gid=*value*** establishes the GID for each file of the file system (msdos, hpfs)
- grpuid** new files receive the same GID as the directory in which they are created (ext2)
- nocheck** equivalent to *check=none* (ext2)
- nogrpuid** new files receive the GID of the creating process, as in System V (ext2, default)
- norock** turns off Rockridge extensions, ending case sensitivity and long file names (iso9660)
- quiet** suppresses corresponding error messages on attempts to execute the commands *chmod* and *chown* (msdos)
- sb=*value*** uses an alternative superblock at the specified block position (normally at positions 1, 8193, 16385, ...) (ext2)
- sysvgroups** see *nogrpuid*
- uid=*value*** determines the GID for each file in the file system (msdos, hpfs)
- umask=*value*** determines the umask for files (msdos, hpfs) 1
- r** file system is mounted as read-only
 - t *type*** mounts a file system of a certain *type* (default: minix; possible values: minix, ext, ext2, xiafs, msdos, hpfs, proc, nfs, iso9660, sysv, xenix, coherent)
 - v** outputs verbose messages

mv [options] path target

Moves files and directories or renames them. If the target already exists and it is a file, it is overwritten; if it is a directory, the specified files and directories are moved into the existing directory. If the `target` does not exist, then only a file or a directory can be specified as the source, and it is renamed to the `target` name.

Available options

- b** creates a backup of a file before overwriting it
 - f** does not ask for confirmation before overwriting files
 - i** asks for confirmation before overwriting files
 - u** moves a file only if it is newer than a target file of the same name
 - S** see `cp` command
 - V** see `cp` command
-

nice [-n value | -value] commands [arguments]

Executes commands with a higher nice-level, i.e., a lower priority. `nice` is usually integrated in the shell. The maximum nice-level is 19. The system administrator can also specify negative values to -20. A default nice-level of 10 is used in lieu of a specification.

nm [options] files

Outputs the symbol table of object files or libraries.

nohup command [arguments] &

This command is usually integrated into the shell. It prevents termination of the shell when the specified `command` terminates.

nroff [options] files

Formats files that contain corresponding format statements for output on the screen or the printer (see also `groff`).

mv

nice

nm

nohup

nroff

openwin

openwin

Shell script to start the X11 environment.

passwd

passwd [*user*]

Changes the user's own password. The system administrator can also change the passwords of other *users*.

pr

pr [*options*] [*files*]

Prepares text *files* for printing. The *file* contents are prepared pagewise and provided with a title line containing the date, the file name, and the page number.

Available options

- +page** begins printing starting with the specified *page*
- column** produces multiple-*column* output
- a** prints columns alongside rather than under one another
- c** outputs nonprintable characters in “^” notation
- d** double-spaced printout
- e[chars[width]]** replaces any number of *characters* with a number of blanks. The default value for tab characters is 8 blanks.
- f** produces a form feed at the end of a page instead of generating a series of blank lines
- h text** replaces the file name in the title line with the specified *text*
- i[chars[width]]** reverse effect of option **-e**
- l length** determines the *length* of a page (default 66 lines)
- n[chars[width]]** outputs a sequential number before each line; optionally, a *character* that separates the number from the text and the *width* of the number can be specified
- o width** creates a left margin of specified *width*
- r** suppresses error messages for files that cannot be opened
- t** suppresses the header and footer
- v** outputs nonprintable characters in format
- w** specifies page width in characters (default 72)

-x displays the processes that are not assigned to a terminal

ps [options]

Outputs a list of currently active processes.

Available options

- a** displays the processes of all users
- h** suppresses the header line
- j** outputs the process' group ID and session ID
- l** verbose output format
- m** provides an overview of storage allocation
- r** lists only currently running processes
- s** provides information on signal status
- u** outputs the name of the process owner and the start time
- w** suppresses the truncation of command lines for wide output

pwd

Outputs the complete path of the current directory

r**cp** [options] sources target

Copies files between computers. The `sources` and the `target` are in the form `user@host:path`, whereby `user@` can be omitted, in which case the current user name is used. For local files only the path is specified.

Available options

- r** recursively copies subdirectories and their contents
- p** retains the file attributes (date, permissions) during copying

rlogin [options] host

Similar to telnet, this command provides a connection to the specified `host` and logs in there. If the current user is entered on the

ps

pwd

rcp

rlogin

remote host in the files `.rhosts` or `/etc/hosts.equiv`, then no password is required.

Available options

-l *name* uses *name* as the user name on the remote host

rm

rm [options] files

Removes one or more files. Removing a file requires write permission in the containing directory. If the file is write protected, confirmation is required. Directories are removed with `rmdir`.

Available options

- f** removes files, even if write protected, without confirmation
- i** asks for confirmation for each file
- r** recursively removes subdirectories and their contents
- v** displays each file name on removal

rmdir

rmdir [options] directories

Removes subdirectories. A directory must be empty to be removed. Alternatively, `rm` can be used with option `-r`, which removes subdirectories and their contents.

rsh

rsh [options] host [commands]

Executes commands on a remote host. Access must be permitted via an entry in `/etc/hosts.equiv` or `~/.rhosts`.

Available options

- l *user*** attempts to execute the specified command under another user name
- n** redirects the standard input device to `/dev/null` (works around problems with `csh`)

Example

```
rsh -luhl zeus.demo.de ls
    executes the command ls as user uhl on the host
    zeus.demo.de
```

```
sdiff [options] file1 file2
```

Compares two files and outputs the differences in two columns (also see `diff`). This output is easier to read than that of `diff`. Lines that are not contained in one of the two files are marked with "<" or ">." A pair of lines that differ are indicated with "|."

sdiff

```
sed [options] [files]
```

Modifies without interaction with the user. This command is usually used in shell scripts to replace, delete or insert text. If no `file` is specified, then `sed` works with the standard input device.

sed

Available options

- e 'statements'** executes the edit statements with the specified files
- f script file** reads the edit statements from *script file*
- n** suppresses the echo of input lines on the screen

```
shutdown [options] time [message]
```

Changes the run level of the system or terminates the system. A `time` and a warning `message` can be passed as arguments. For an immediate shutdown, the time `now` is specified.

shutdown

Available options

- c** interrupts a shutdown in progress
- h** halts the system with the termination of all processes and unmounts the file systems
- k** does not execute a shutdown, but only displays the warning
- r** reboots the system

-t *sec* delay in seconds between the display of the warning message and sending the kill signals

sleep *time*

sleep

Sleeps the specified *time* in seconds. This command is usually used in shell scripts.

sort [*options*] [*files*]

sort

Sorts the lines in the specified *files*. If no *files* are specified, the standard input device is processed.

Available options

- +n-m** sets the sorting key between fields *n* and *m*
- b** suppresses leading blanks
- c** checks whether the specified files are already sorted; if so, the program terminates with an error message
- d** ignores punctuation marks during sorting
- f** case-sensitive (upper/lower)
- i** ignores nonprintable ASCII characters
- m** mixes two input *files*
- M** interprets the first three characters as a month name (JAN, FEB, . . . , DEC) and sorts by month
- n** sorts numerically
- o *file*** redirects the standard output device to a file
- r** inverts sorting order
- t *char*** specifies the delimiting character for columns (default: blank or tab)
- u** removes duplicate lines

Example

```
sort +2n -t:/etc/passwd
    sorts the password file numerically according to the third
    column
```

strings [options] files

Searches for character strings in binary and object files or programs. A character string is considered to be any sequence of four or more printable characters terminated with null.

strings

Available options

- a** normally for object files, only the code and data segments are searched; this option assures that the whole file is processed
- f** each character string is preceded by the corresponding file name
- n** specifies the minimum length of the character string (default: 4)
- o** outputs the position of a character string in bytes

strip [options] files

Strips symbol, debug, line numbers, and other information from object files and programs, thus reducing their size.

strip

stty [options] [modes]

Sets terminal IO modes. This includes all general settings of the terminal as well as speed and handshaking and the function of special characters. A list of all possible settings is displayed using the option `--help`.

stty

Available options

- a** displays all current settings
- help** displays help text

su [-] [user] [arguments]

Starts a new shell as a different user. This program is used to log in on a terminal that is already being used by another user. Omitting

su

the user opens a root shell. The new shell is terminated by entering `exit` or `<Ctrl-d>`. If `"-"` is specified as option, then the complete login process is run on opening the new shell. In addition, the option `-c` allows execution of commands under a different user ID.

```
tail [options] [files]
```

tail

Outputs the last 10 lines of the specified files.

Available options

- c[b|k|m]** outputs the last *n* bytes in blocks (*b*), kilobytes (*k*), or megabytes (*m*)
- f** does not terminate after outputting the last lines, but waits until the file is written to. As soon as new lines are appended to the file, they are output. In this mode the program is terminated with `break (<Ctrl-c>)`. This mode proves especially suitable for monitoring log files
- n** outputs the last *n* lines
- v** outputs the file name as title line

```
talk user[@host] [tty]
```

talk

Sets up a `talk` connection to the specified user. If this user is logged in on multiple terminals, then the terminal to be used can be entered in the command line. A `talk` connection splits the terminal screen in two parts, with local input shown in the top half and remote input in the lower half. The connection is terminated with `<Ctrl-c>`. Unfortunately, there are two incompatible versions of `talk`, so that connection to a different platform does not always succeed.

```
tar [options] [archive] [files]
```

tar

Manages tar archives (originally on magnetic tape). This command writes files to an archive or reads them from an archive. At least one of the following operations must be passed as parameter.

Available operations

- c** creates the archive
- r** appends files to an archive (not on tape)
- t** outputs the contents of an archive
- u** appends files to an archive if they are not already contained or if they have been modified (not on tape)
- x** extracts files from an archive

Additional options

- b *n*** sets the blocking factor to *n*
- f *archive*** specifies the *archive*, which can be a normal file or a device file such as `/dev/rmt0` for a tape drive or `/dev/fd0` for a diskette
- h** archives referenced files instead of their symbolic links
- k** prevents overwriting of existing files
- L** follows symbolic links
- m** sets the modification time on extracting files to the current time
- M** creates or extracts from a multivolume archive, which can encompass multiple diskettes or tapes
- N *date*** archives only files that are newer than the specified *date*
- o** sets the owner on extracting files to the current user
- O** extracts files to the standard output device
- v** displays the file name on archiving or extracting
- z** compresses the archive on creation and decompresses on extraction

Examples

```
tar -cvf archive.tar *
    saves all files and subdirectories of the current directory in
    an archive named archive.tar
```

```
tar -cvf /dev/fd0 *.txt
    saves all files in the current directory with the extension
    .txt from the diskette in the first floppy disk drive
```

```
tar -xvfb20 /dev/rmt0
    extracts all files from the first tape drive (block size 20)
```

```
tar -tvf zarchive.tar.z
```

lists the contents of a compressed tar archive

tee

```
tee [options] [files]
```

This program is used as a filter. It copies the standard input device to the standard output device and the specified `files`.

Available options

- a** appends the data received from the standard input device to the end of the `file` instead of overwriting itself
 - i** ignores interrupt signals
-

telnet

```
telnet [host [port]]
```

Opens a connection to the specified `host` using the telnet protocol. A `port` number can be specified optionally. This program is often used to test services that are available for connections on certain ports. If no `host` is specified, then `telnet` goes into command mode, where `telnet` commands can be entered. The command `help` lists all important commands.

test

```
test condition
```

Evaluates the specified `condition` and returns zero if the result is true, else a nonzero value. Alternatively, the `condition` can be put in square braces, which is primarily used in shell scripts.

Files

- b file** *file* is a block device
- c file** *file* is a character device
- d file** *file* is a directory
- f file** *file* is a normal file
- g file** set group ID bit (SGID) of *file* is set
- G file** effective GID matches owner's group
- k file** sticky bit of *file* is set
- O file** effective UID matches the file owner

-p file *file* is a named pipe
-r file *file* exists and is readable
-s file *file* is larger than 0 bytes
-S file *file* is a socket
-t [n] file descriptor *n* corresponds to a terminal
-u file set user ID bit (SUID) of *file* is set
-w file *file* exists and is writable
-x file *file* exists and is executable
d1 -ef d2 *files d1* and *d2* are linked
d1 -nt d2 *file d1* is newer than *file d2*
d1 -ot d2 *file d1* is older than *file d2*

Character strings

-n z1 length of character string *z1* is greater than zero
-z z1 length of character string *z1* is zero
z1 character string *z1* is not null
z1=z2 *z1* is equal to *z2*
z1 != z2 *z1* is not equal to *z2*
z1 < z2 *z1* is lexicographically smaller than *z2*
z1 > z2 *z1* is lexicographically greater than *z2*

Numeric conditions

n1 -eq n2 *n1* equals *n2*
n1 -ge n2 *n1* is greater than or equal to *n2*
n1 -gt n2 *n1* is greater than *n2*
n1 -le n2 *n1* is less than or equal to *n2*
n1 -lt n2 *n1* is smaller than *n2*
n1 -ne n2 *n1* is not equal to *n2*

Combinatoric conditions

! a1 true if expression *a1* is false
a1 -a a2 true if *a1* and *a2* are true
a1 -o a2 true if *a1* or *a2* is true

Examples

```
if [ -f /etc/shadow ]
    tests whether file /etc/shadow exists
```

```
if [ "$res" != "j" ]
```

Does the content of the variable `res` equal "j"?

```
while [ -z "$res" ]
```

Does the variable `res` contain an empty string?

time

Executes the specified command and then displays the execution time.

```
touch [options] files
```

touch

Changes the last access date/time and the last modification date/time of files. If a specified file does not exist, it is created as empty.

Available options

- a** changes only the time of the last access
 - c** disables the creation of empty files for nonexistent ones
 - m** changes only the time of the last modification
 - r file** transfers the time from a specified reference *file*
 - t value** sets the file date and the system time to the specified *value* with the format MMDDhhmm (month, day, hour, minute)
-

```
tr [options] [string1 [string2]]
```

tr

Copies the standard input device to the standard output device and in the process replaces or deletes characters. If a character from `string1` is found in the standard input device, then it is replaced with the corresponding character from `string2`.

Available options

- c** outputs the complement of the set of characters in `string1`
- d** deletes characters that appear in `string1`
- s** suppresses repeated sequences in the output

troff [options] [files]

Formats files for printer or linotronic machine (also see `nroff` and `groff`)

troff

true

This command returns only 0 (“successful”) as return code. It is used primarily in shell scripts.

true

umask [value]

Outputs the current value of the file generation mask as an octal number or sets this `value`. This mask determines the maximum permissions that a newly created file can receive. Here the `umask` value is subtracted from the permissions of the file to be created.

umask

uname [options]

Outputs the name and version number of the current system.

uname

Available options

- a** outputs all available information
- m** outputs hardware (processor) type
- n** outputs the host name
- r** outputs the version number of the operating system
- s** outputs the name of the operating system
- v** outputs the date and time of compilation of the kernel

uncompress [options] [files]

Restores an original file that was compressed with `compress`.

uncompress

Available options

- c** outputs the file contents to the standard output device. Here `uncompress` behaves like the command `zcat`.

uniq

```
uniq [options] [file1[file2]]
```

Deletes successive identical lines in the linewise sorted `file1` and outputs these to `file2` (or the standard output device).

Available options

-c outputs the number of repetitions
-d outputs only lines that occur redundantly
-u outputs only lines that occur uniquely
-n skips a number *n* of fields (with tabs and blanks as delimiters) before comparing two lines
+n skips *n* characters before beginning to compare
-w specifies the number of characters to be compared

uptime

```
uptime
```

Outputs the current time, the time since the last reboot, the number of logged users, and the momentary system load.

uudecode

```
uudecode [file]
```

Decodes a `file` encoded with `uuencode` using its original name, owner, and permissions

uuencode

```
uuencode [file1] name
```

Encodes binary files so that they can be represented as ASCII files and sent via e-mail. An encoded file is 35% larger than the original. The result is written to the standard output device. The specified `name` corresponds to the file name after the file is unpacked by its recipient.

vi

```
vi [options] [files]
```

Full-screen editor for processing ASCII files. It is largely based on `ex` and generally functions on all terminals.

w [options] [users]

Displays all currently logged `users` and their activities. Without parameters, all users are output; with a name, only the specified user.

w

Available options

- h** suppresses a title line
- f** determines whether the login terminal should also be output
- s** concise output format

wc [options] [files]

Counts the number of characters, words, and lines in a text file.

wc

Available options

- c** only the number of characters
- l** only the number of lines
- w** only the number of words

whatis [commands]

Outputs a short description of the specified commands from the Online Manual

whatis

which [commands]

Outputs the file path of the specified `commands` (usually an internal shell command)

which

who [options] [file]

Outputs a list of currently users currently logged in, their terminals, the login time and the name of the host on which they logged in. If a file name is specified in addition, then this `file` is used for evaluation instead of `/etc/utmp`.

who

Available options

- am i** outputs the user's own data
- i** outputs how long the user was inactive
- H** outputs column headings
- q** outputs only the login name and the number of users
- w** displays whether the user accepts (+) messages generated with `write` or not (-)

write `user` [`terminal`]

write

Outputs a message on a certain user's `terminal`. The message is read by the standard input device until it encounters EOF (<Ctrl-d>).

xargs [`options`] [`commands`]

xargs

Executes a `command` with the (multiple) arguments read from the standard input device. This enables passing lists of any length of arguments to commands.

Available options

- 0** file names are terminated by the character null
- e *string*** ends processing as soon as the specified character *string* appears in the list of file names (default is "_")
- l *n*** executes the command with *n* arguments
- n *n*** executes the command with at most *n* arguments
- p** interactive processing where the user must respond with "y" before a command is executed
- s *n*** each argument may contain at most *n* characters
- t** displays the command before its execution

zcat [`files`]

zcat

Decompresses the specified `files` and outputs their contents to the standard output device. The compressed files remain untouched.

Appendix

A.1 Overview of /etc files

Most configuration files reside in the directory `/etc`. This section gives an overview of the most important of these configuration files. Our list refers primarily to the Slackware distribution, which does not completely comply with the new Linux File System Standard (FSSTND).

- **boot ptab** - configuration file for `bootpd` daemon boot
- **sh.cshrc** - global definitions for `(t)cs`h
- **cs**h.**login** - global login shell definitions for `(t)cs`h login
- **diphosts** - list of machines permitted to make a SLIP connection SLIP
- **DIR_COLORS** - configuration file with color settings of the `ls` command ls colors
- **disktab** - file for determination of nonstandard hard disk parameters for LILO hard disk
- **exports** - defines the directories to be exported by NFS NFS
- **fdprm** - parameters for floppy disk drive s floppy disk
- **fstab** - contains the file system s and swap partition s to be mounted or activated on booting file system & swap
- **ftp access** - configuration file for the FTP daemon; allows the definition of access restrictions and messages for `wu-ftp`d or `diku-ftp`d FTP
- **ftpusers** - configuration file for the FTP daemon. Users listed in this file cannot log in per FTP
- **gateways** - file with a list of router s
- **gettydefs** - configuration file for `getty`

groups	• group – This file lists all groups and their member users. If a user belongs to multiple groups, then this user name appears with multiple groups. The only group name contained in the <code>passwd</code> file is the user's primary group name
shadow	• gshadow – shadow file for the file <code>group</code>
address resolution	• host.conf – specifies the sequence in which to search for an IP address
network hosts	• hosts – This file contains the names and IP addresses of other computers. Depending on the settings in the file <code>host.conf</code> , this file is searched before or after the name server to find the IP address of a host
	• hosts.allow – lists computers with permission to access local network services (tcp wrapper)
	• hosts.deny – lists computers that are denied access to local network services (tcp wrapper)
trusted users	• hosts.equiv – defines trusted users/hosts for services with the Berkeley r-utilities
printers	• hosts.lpd – lists computers with permission to use local printers
Internet	• inetd.conf – This configuration file for the daemon <code>inetd</code> establishes a one-to-one correspondence between a connection to a certain port and the daemon to be started
	• inittab – This configuration file for the <code>init</code> process contains the assignments of shell script <code>s</code> to run level <code>s</code>
run level	
login	• issue – This text file is displayed before the login prompt. Normally it contains the name of the host and a greeting message
dynamic linker	• ld.so.cache – file with configuration data for dynamic linker
shared libraries	• ld.so.conf – file with paths to shared libraries
LILO	• lilo.conf – configuration file for Linux Loader (LILO).
file type	• magic – This file establishes the correspondence between byte patterns at the beginning of a file and the respective file type. A file's type can be determined by searching <code>magic</code> for the file's starting byte pattern
login message	• motd – This text file (message of the day) is normally automatically displayed after each login
file systems	• mtab – This internal table for <code>mount</code> lists all currently mounted file system <code>s</code>

- **mttools** - configuration file for MTools
- **named.boot** - boot file for the name server name server
- **networks** - contains the IP addresses and names of known networks
- **passwd** - User s are defined in this file by specifying each user's ID, user name, login shell and primary group. The actual passwords are stored in the file **shadow**, which, unlike the **passwd** file, can only be read with **root** permissions users and passwords
- **printcap** - configuration file for printer queue s printer queue
- **profile** - global profile file for Bourne shells (**bash**)
- **protocols** - defines the TCP/IP packet types TCP/IP
- **resolv.conf** - configuration file for the TCP/IP system containing the local domain name and the address of the name server
- **rpc** - configuration file for RPC server RPC
- **securetty** - specifies on which TTYs the superuser can log in
- **services** - contains the assignment of TCP/IP ports to corresponding names of services (programs) port assignment
- **shadow** - file with encrypted user passwords
- **shells** - This file contains the valid system shells. When the user logs in as **anonymous** or **ftp**, the FTP daemon checks whether the user's login shell is entered in this list. If not, the login is denied
- **syslog.conf** - configuration file for the **syslog** daemon syslog
- **termcap** - configuration file for the **termcap** library defining the control characters of the various types of terminals
- **utmp**, **wtmp** - log file of all user logins and logouts
- **xmmounttab** - configuration file for **xmmount**
- **xvmounttab** - configuration file for **xvmount**

A.2 Overview of /etc directories

- **/etc/default** - stores the default values of parameters:
 - **useradd** - default values for **useradd** command new user
 - **getty.XXX** - configuration of **getty** program for port port
XXX

	<ul style="list-style-type: none"> - uugetty.XXX - configuration of uugetty program for port XXX
LILO	<ul style="list-style-type: none"> • lilo - contains LILO installation program
new user	<ul style="list-style-type: none"> • skel - contains files that, on execution of <code>useradd</code>, are automatically copied into the home directory of a new user
run level	<ul style="list-style-type: none"> • rc.d - contains scripts that are invoked by a process on transition to the respective run level s:
shutdown	<ul style="list-style-type: none"> - rc.0 - script for run level 0 (system shutdown) - rc.6 - script for run level 6 (login via <code>xdm</code>)
single-user mode	<ul style="list-style-type: none"> - rc.K - script for single-user mode
multi-user mode	<ul style="list-style-type: none"> - rc.M - script for multi-user mode (run levels 1 to 6)
consistency	<ul style="list-style-type: none"> - rc.S - script that activates swapping and checks consistency of file system on booting
keyboard	<ul style="list-style-type: none"> - rc.keymap - script for loading country-specific keyboard layouts
daemon	<ul style="list-style-type: none"> - rc.local - script to start local daemon s
network	<ul style="list-style-type: none"> - rc.inet1 - script to initialize network interfaces - rc.inet2 - script to start network daemons
serial	<ul style="list-style-type: none"> - rc.serial - script to configure serial interfaces

A.3 Configuration of the kernel

The following listing shows an exemplary configuration of the Linux kernel. Since the kernel remains in a state of flux, deviations are possible.

```
hermes:/usr/src/linux# make config

*
* General setup
*
Kernel math emulation (CONFIG_MATH_EMULATION) [n] n
Normal floppy disk support (CONFIG_BLK_DEV_FD) [y] y
Normal harddisk support (CONFIG_BLK_DEV_HD) [y] y
XT harddisk support (CONFIG_BLK_DEV_XD) [n] n
Networking support (CONFIG_NET) [y] y
Limit memory to low 16MB (CONFIG_MAX_16M) [n] n
PCI bios support (CONFIG_PCI) [n] y
System V IPC (CONFIG_SYSVIPC) [y] y

Kernel support for ELF binaries (CONFIG_BINFMT_ELF) [y] y
Use -m486 flag for 486-specific optimizations (CONFIG_M486) [y] y
*
* Networking options
*
TCP/IP networking (CONFIG_INET) [y] y
IP forwarding/gatewaying (CONFIG_IP_FORWARD) [n] y
IP multicasting (ALPHA) (CONFIG_IP_MULTICAST) [n] n
IP firewalling (CONFIG_IP_FIREWALL) [n] n
IP accounting (CONFIG_IP_ACCT) [n] n
```

```

*
* (it is safe to leave these untouched)
*
PC/TCP compatibility mode (CONFIG_INET_PCTCP) [n] n
Reverse ARP (CONFIG_INET_RARP) [n] n
Assume subnets are local (CONFIG_INET_SNARL) [y] y
Disable NAGLE algorithm (normally enabled) (CONFIG_TCP_NAGLE_OFF) [n] n
The IPX protocol (CONFIG_IPX) [n] y
*
* SCSI support
*
SCSI support? (CONFIG_SCSI) [n] y
*
* SCSI support type (disk, tape, CDrom)
*
Scsi disk support (CONFIG_BLK_DEV_SD) [y] y
Scsi tape support (CONFIG_CHR_DEV_ST) [n] y
Scsi CDROM support (CONFIG_BLK_DEV_SR) [n] y
Scsi generic support (CONFIG_CHR_DEV_SG) [n] n
*
* SCSI low-level drivers
*
Adaptec AHA152X support (CONFIG_SCSI_AHA152X) [n] n
Adaptec AHA1542 support (CONFIG_SCSI_AHA1542) [y] y
Adaptec AHA1740 support (CONFIG_SCSI_AHA1740) [n] n
Adaptec AHA274X/284X support (CONFIG_SCSI_AHA274X) [n] n
BusLogic SCSI support (CONFIG_SCSI_BUSLOGIC) [n] n
UltraStor 14F/34F support (CONFIG_SCSI_U14_34F) [n] n
Future Domain 16xx SCSI support (CONFIG_SCSI_FUTURE_DOMAIN) [n] n
Generic NCR5380 SCSI support (CONFIG_SCSI_GENERIC_NCR5380) [n] n
NCR53c7,8xx SCSI support (CONFIG_SCSI_NCR53C7XX) [n] y
Always IN2000 SCSI support (test release) (CONFIG_SCSI_IN2000) [n] n
PAS16 SCSI support (CONFIG_SCSI_PAS16) [n] n
QLOGIC SCSI support (CONFIG_SCSI_QLOGIC) [n] n
Seagate ST-02 and Future Domain TMC-8xx SCSI support
(CONFIG_SCSI_SEAGATE) [n] n
Trantor T128/T128F/T228 SCSI support (CONFIG_SCSI_T128) [n] n
UltraStor SCSI support (CONFIG_SCSI_ULTRASTOR) [n] n
7000FASST SCSI support (CONFIG_SCSI_7000FASST) [n] n
EATA ISA/EISA (DPT PM2011/021/012/022/122/322) support (CONFIG_SCSI_EATA)
[n] n
*
* Network device support
*
Network device support? (CONFIG_NETDEVICES) [y] y
Dummy net driver support (CONFIG_DUMMY) [n] n
SLIP (serial line) support (CONFIG_SLIP) [n] y
CSLIP compressed headers (SL_COMPRESSED) [y] y
16 channels instead of 4 (SL_SLIP_LOTS) [n] n
PPP (point-to-point) support (CONFIG_PPP) [n] y
PLIP (parallel port) support (CONFIG_PLIP) [n] n
Load balancing support (experimental) (CONFIG_SLAVE_BALANCING) [n] n
Do you want to be offered ALPHA test drivers (CONFIG_NET_ALPHA) [n] n
Western Digital/SMC cards (CONFIG_NET_VENDOR_SMC) [n] y
WD80*3 support (CONFIG_WD80X3) [n] n
SMC Ultra support (CONFIG_ULTRA) [n] y
AMD LANCE and PCnet (AT1500 and NE2100) support (CONFIG_LANCE) [n] n
3COM cards (CONFIG_NET_VENDOR_3COM) [y] y
3c501 support (CONFIG_EL1) [n] n
3c503 support (CONFIG_EL2) [n] n
3c509/3c579 support (CONFIG_EL3) [y] y
Other ISA cards (CONFIG_NET_ISA) [n] n
EISA, VLB, PCI and on board controllers (CONFIG_NET_EISA) [n] n
Pocket and portable adaptors (CONFIG_NET_POCKET) [n] y
AT-LAN-TEC/RealTek pocket adaptor support (CONFIG_ATP) [n] n
D-Link DE600 pocket adaptor support (CONFIG_DE600) [n] n
D-Link DE620 pocket adaptor support (CONFIG_DE620) [n] y
*
* CD-ROM drivers
*
Sony CDU31A/CDU33A CDROM driver support (CONFIG_CDU31A) [n] n
Mitsumi CDROM driver support (CONFIG_MCD) [n] n
Matsushita/Panasonic CDROM driver support (CONFIG_SBPCD) [n] n
*
* Filesystems
*
Standard (minix) fs support (CONFIG_MINIX_FS) [y] y
Extended fs support (CONFIG_EXT_FS) [n] n
Second extended fs support (CONFIG_EXT2_FS) [y] y
*
xiafs filesystem support (CONFIG_XIA_FS) [n] n
msdos fs support (CONFIG_MSDFS) [y] y
umsdos: Unix like fs on top of std MSDOS FAT fs (CONFIG_UMSDOS_FS) [n] n
/proc filesystem support (CONFIG_PROC_FS) [y] y
NFS filesystem support (CONFIG_NFS_FS) [y] y

```

```
ISO9660 cdrom filesystem support (CONFIG_ISO9660_FS) [y] y
OS/2 HPFS filesystem support (read only) (CONFIG_HPFS_FS) [n] n
System V and Coherent filesystem support (CONFIG_SYSV_FS) [n] n
*
* character devices
*
Cyclades async mux support (CONFIG_CYCLADES) [n] n
Parallel printer support (CONFIG_PRINTER) [n] y
Logitech busmouse support (CONFIG_BUSMOUSE) [n] n
PS/2 mouse (aka "auxiliary device") support (CONFIG_PSMOUSE) [n] y
C&T 82C710 mouse port support (as on TI Travelmate) (CONFIG_82C710_MOUSE)
[y] y
Microsoft busmouse support (CONFIG_MS_BUSMOUSE) [n] n
ATIIXL busmouse support (CONFIG_ATIIXL_BUSMOUSE) [n] n
Selection (cut and paste for virtual consoles) (CONFIG_SELECTION) [n] n
VESA Power Saving Protocol Support (CONFIG_VESA_PSPM) [n] n
QIC-02 tape support (CONFIG_QIC02_TAPE) [n] n
QIC-117 tape support (CONFIG_FTAPE) [n] n
*
* Sound
*
Sound card support (CONFIG_SOUND) [n] n
*
* Kernel hacking
*
Kernel profiling support (CONFIG_PROFILE) [n] n
Verbose scsi error reporting (kernel size +=12K) (CONFIG SCSI_CONSTANTS)
[y] y

The linux kernel is now hopefully configured for your setup.
Check the top-level Makefile for additional configuration,
and do a 'make dep ; make clean' if you want to be sure all
the files are correctly re-made

hermes:/usr/src/linux#
```

A.4 Further reading

Andeleigh, Prabhat K.

UNIX System Architecture. Prentice-Hall [1993]

Carl-Mitchell, Smoot, and Quarterman, John S.

Practical Internetworking with TCP/IP and UNIX.

Addison-Wesley [1993]

Comer, Douglas E.

Internetworking with TCP/IP, Vol. 1-3, Prentice-Hall [1991]

Flanagan, David

X Toolkit Intrinsics Reference Manual, Vol. 5. O'Reilly [1993]

Gilly, Daniel

UNIX in a Nutshell. O'Reilly [1992]

Gulbins, Jürgen

UNIX. Springer [1988]

Heller, Dan

XView Programming Manual, Vol. 7. O'Reilly [1991]

Motif Programming Manual, Vol. 6. O'Reilly [1992]

Hewlett-Packard

Ultimate Guide to the Vi and Ex Text Editors.

Addison Wesley [1990]

Kehoe, Brendan P.

Zen and the Art of the Internet. Prentice-Hall [1993]

Krol, Ed

The Whole Internet: User's Guide and Catalog. O'Reilly [1993]

Lippmann, Stanley B.

C++ Primer. Addison Wesley [1991]

Mui, Linda and Pearce, Eric

X Window System Administrator's Guide, Vol. 8. O'Reilly [1993]

Nemeth E. Snyder G., Seebass S., Trent, R. H.

UNIX System Administration Handbook, 2nd Ed.

Prentice Hall [1995]

Nye, Adrian

Xlib Programming Manual, Vol. 1. O'Reilly [1992]

Xlib Reference Manual, Vol. 2. O'Reilly [1992]

Nye, Adrian, and O'Reilly, Tim

X Toolkit Intrinsics Programming Manual, Vol. 4.

O'Reilly [1990]

Open Software Foundation

OSF/Motif Programmer's Guide. Prentice-Hall [1993]

OSF/Motif Programmer's Reference. Prentice-Hall [1993]

OSF/Motif Users' Guide. Prentice-Hall [1993]

Quercia, Valerie, and O'Reilly, Tim

X Window System User's Guide, Vol. 3. O'Reilly [1991]

Schoonover, Michael A.

GNU Emacs—UNIX Text Editing and Programming.

Addison-Wesley [1992]

Stevens, W. Richard

Advanced Programming in the UNIX Environment.

Addison Wesley [1992]

UNIX Network Programming. Prentice-Hall [1990]

Tanenbaum, Andrew S.

Distributed Open Systems. Prentice-Hall [1995]

Young, Douglas A.

X Window System, Programming and Applications with Xt.

Prentice-Hall [1990]

Index

—A—

Abacus, 67
 Accelerated X, 154
 accelerator board, 154
 access path, 26
 access permissions, 242
 access privileges, 11, 41
 access time, 72
 Ada, 326
 Ada compiler, 128
 Ada-mode, 321
 Ada9X, 326
 adaptation, 750
 Adaptec, 76
 Adaptec controller, 107
 address, 411
 Address Resolution Protocol
 (ARP), 186
 ADDSWAP, 87
 Adlib, 77
 administration, 119
 administrator, 11, 119
 Adobe fonts, 273
 Advanced Research Project
 Agency, 182
 AFS, 212
 algorithm, 269
 alias, 20, 26, 201
 <Alt>, 158
 alternative shells, 44
 <AltGr>, 158
 Amsterdam, University of, 328
 Andrew applications, 278
 Andrew file system, 212
 Andrew User Interface System
 (AUIS), 278
 animations, 278
 anonymous, 125, 132
 ANSI SQL standard, 279
 ANSI standard, 280
 answering machine, 114
 APL, 325
 append, 96, 103, 106, 311
 appendfile, 242
 Apple, 67, 280, 337
 Quicktime format, 273
 Apple Finder, 152
 Apple Macintosh, 220
 application defaults, 169
 application level, 183
 applications, 72, 169, 265
 apropos, 256, 353
 aps, 113
 ar, 353
 Archie server, 230
 archive, 299, 362, 400
 archiving, 351
 Arcnet, 77, 181
 Arena, 251
 argument, 304, 313
 arithmetic coprocessor, 106
 emulation, 106
 ARP broadcast, 186, 188
 ARP reply, 186, 188

ARPA, 182
ARPANET, 182
array, 301, 302, 312
arrayp, 310
ASCII, 268
 environment, 267
 format, 275
 sequence, 323
 terminals, 77, 287
 text, 274
asedit, 268
Assembler, 102
associative array, 330, 338
at, 34, 354
AT, 75
AT bus, 76
AT bus controller, 117
AT&T, 6, 151
Athena text widgets, 267
Athena Project, 143
Athena widget set, 150
atq, 355
atrm, 355
attribute, 172
 driver specific, 240
 generic, 240
attributes, 242
authentication, 194, 219
authentication protocol, 197
auto-mode-alist, 321
autoexec.bat, 53, 60
autoload, 321
automatic activation, 101
automatic backup copy, 287
automatic connection, 209
automatic detection, 161, 162
automatic error correction, 135
automatic extension, 45, 46
automatic mount, 99
automation, 328
awk, 338, 355

—B—

background, 34
Backquoting, 26
backup, 124, 138, 400
 on floppy disks, 133
backups, restoring, 133
bandwidth, 159
basename, 356
bash, 21, 44, 356
bashrc, 23, 125
Basic, 325
basic file system, 82
batch, 356
bc, 356
Berkeley, 4, 6
Berkeley Distribution, 231
Berkeley r-utilities, 410
Berkeley sockets, 200
Berkeley, University of
 California at, 20
Bezier curves, 271
bin, 21, 124, 132
binary mode, 43
binding, 308
bindings, 177
BIOS, 49, 50
black screen, 57
BLOB (Binary Large Object),
 280
block, 288
block devices, 19
Bond University, Australia,
 279
boot diskette
boot disk, 53, 54, 72, 78, 79,
 82, 92, 93, 104–106,
 124, 137, 138, 141,
 369
 creating, 90
 custom, 79
boot drive, 58
boot image, 59

boot image file, 51
 boot kernel, 90
 boot manager, 40, 78, 80, 93,
 97
 boot messages, 81
 boot options, 94
 boot parameters, 81
 boot prompt, 106
 boot sector, 94, 141
 bootable disk, 79
 booting, 50, 93, 94, 103, 104,
 106, 123–125, 127,
 138, 141
 second hard disk, 93
 bootoff, 60
 booton, 60
 bootpd, 223
 Borland, 343
 Bourne Again shell (bash), 21
 Bourne shell, 20, 348, 411
 Bradley, J., 269
 branches, 29
 Braunschweig, 72
 breakpoint, 338–340, 347
 browser, 268, 320
 BSD UNIX, 6, 200, 231
 buffe, 289
 buffer, 123
 buffer (datatype), 302
 bufferp, 310
 bulletin board, 245
 bus mouse, 76, 107
 bus system, 76
 byte-code function, 302

—C—

C, 102, 111, 184, 201, 206,
 210, 268, 279, 325,
 332, 339, 341
 ANSI, 326
 K&R, 326
 Objective C, 326
 C compiler, 8, 69, 75, 325,
 326, 358
 GNU C, 8
 Objective C, 8
 C library, 129, 136, 140, 329
 C news, 248
 C programs, 299
 C shell, 20, 46
 C++, 153, 154, 268, 279, 280,
 325, 326, 335, 339
 c.o.l.a., 3
 cache, 123
 CAD, 144
 cal, 357
 calculator, 64
 calendar, 357
 callback, 335
 Caltech Electronic CAD
 Distribution, 283
 CAP, 220
 car, 310
 Card, Remy, 38
 Carnegie Mellon University,
 276
 carriage return, 42, 111
 case, 30
 cat, 40, 357
 cc, 358
 cd, 358
 CD subscription, 71
 CD-ROM, 39, 69–71, 77, 78,
 81, 88
 running from, 72
 CD-ROM drive, 116, 117
 CDE, 7, 152
 cdr, 310
 central scripts, 238
 CFLAGS, 346
 channel, 24, 248, 261
 character, 301
 character conversion, 111
 character devices, 19

- character sets, 129
 - international, 111
- chat, 194, 195
- chat and pppd, 196
- check in, 343
- check out, 343
- checksum, 360
- Chen, Young, 335
- chfn, 131
- chgrp, 358
- chipset, 76
- chmod, 358
- chown, 359
- chsh, 132
- ci, 343
- Circuitry simulation, 283
- cksum, 360
- class, 169, 171, 172
- class C network, 185
- clear, 360
- client/server architecture, 144
- Clipper, 278
- clisp, 328
- clock, 161–163
- CLOS, 328
- cmp, 360
- co, 343
- code character, 313
- COFF, 62
- color, 174, 288, 295, 317
- color tables, 129
- Columbia Appletalk Packet (CAP), 220
- comm, 360
- |Command, 237
- command description, 407
- command history, 46
- command shell, 15
- commands, 35, 124, 128, 255
 - cat, 35
 - exit, 35
 - grep, 36
 - kill, 36
 - man, 35
 - mkdir, 35
 - more, 35
 - passwd, 36
 - ps, 36
 - rmdir, 35
 - su, 36
- comment, 308
- Common Desktop Environment, 7, 152
- Common Lisp, 315, 328
- communication, 148
- comp.os.linux, 245, 259
- comp.os.linux.announce, 3
- comp.unix.questions, 260
- compact, 94
- compare, 360, 370, 371, 397
- compilation, 90, 96, 103, 104, 346, 349, 387
 - kernel, 102
 - time required, 91
- compiler options, 349, 350
- Compose, 158
- compress, 48, 348, 361, 405
- compression, 104, 348, 361, 379
- computer algebra, 280
- concat, 312
- concatenation, 35
- cond, 305
- condition-case, 315
- conf, 124
- config, 244
- config.sys, 53
- configuration, 90, 99, 129
 - Emacs, 314
 - file, 99, 106, 109
 - kernel, 103
 - keyboard layout, 102
 - LILO, 94
 - modification, 99
 - options, 103

printer, 109, 110
 script, 103
 window manager, 173
 configuration files, 124, 131
 configuration parameters, 90
 configure, 349
 connection, 190
 cons, 310
 cons cell, 310, 324
 consistency, 412
 consistency check, 134
 console, 108
 consp, 310
 continuation, 110
 continuous display, 339, 340
 control construct, 330
 control elements, 148
 converter, 327
 copies printed, 110
 coprocessor, 106
 emulation, 106
 coprocessor emulator, 103
 copy, 361, 362
 copy-face, 318
 copying a file, 35
 CORBA, 153
 core, 14
 COSE Initiative, 7, 152
 count, 407
 cover sheet, 109
 cp, 139, 361
 cpio, 362
 CPU, 13, 58
 crash, 134
 crond, 109
 crontab, 34, 209, 364
 Crossfire, 284
 csh, 364
 cshrc, 20, 125
 csplit, 365
 ctags, 366
 <Ctrl-Alt>, 166
 <Ctrl-Alt +>, 162

<Ctrl-Alt ->, 162
 <Ctrl-Alt-Backspace>, 156,
 157, 166
 <Ctrl-g>, 317
 <Ctrl-h><Ctrl-f>, 308
 <Ctrl-h><Ctrl-v>, 308
 <Ctrl-h v>, 317
 <Ctrl-x> <Ctrl-f>, 345
 <Ctrl-x> <Ctrl-s>, 345
 ctwm, 150
 cu, 206
 cursor, 312
 position, 296
 cursor keys, 45
 cursorColor, 172
 custom boot disk, 80, 90
 customizing, 103, 148
 cut, 367

—D—

Döring, Gert, 114
 daemon, 108, 123, 194, 202,
 410, 412
 FTP, 409, 411
 Internet, 225
 log file, 109
 mail, 237
 message from, 108
 mount, 219
 network, 120
 output from, 108
 PC/NFS, 218
 portmap, 211, 212
 printer, 34, 109
 protocol, 35
 rpc.mountd, 212
 rpc.nfsd, 212
 syslog, 35
 daemons, 12, 34
 damage to monitor, 162, 166
 DARPA NET, 182
 data exchange, 40

- data forwarding, 205
- data loss, 50, 54, 123
- data package, 183
- data packet, 190
- data representation, 210
- data transfer rate, 76
- data type, 328
 - See type
- database, 279
 - of FTP servers, 230
- datagram, 183
- date, 367
- DBSD, 350
- DCE, 212
- dd, 51, 80, 101, 369
- debug, 106
- debug level, 106
- debug mode, 109
- debugger, 347
- debugging, 136
- DEC, 151
- declaration, 301, 350
- DECnet, 146
- decode, 406
- decompress, 80, 405
- decompression, 104, 408
- default major mode, 316
- default route, 187, 188, 196
- default settings, 148, 169, 170
- default-fill-column, 317
- default-tab-width, 317
- default.el, 314
- defaults, 100
- define-key, 322
- defun, 304
- defvar, 304
- delay, 95
- delete files, 396
- delete-region, 312
- delimiter, 15
- deliver, 235, 236, 244
- denied access, 410
- dependencies, 104, 341
- describe-function, 308
- describe-variable, 308
- Deskjet, 113
- desktop manager, 149
- dev, 18, 19, 116, 124, 132, 139
- dev/fd0, 106
- dev/lp1, 18
- dev/mouse, 158
- dev/XOR, 63
- developing environment, 288
- development environment, 337
- device, 14, 18, 49, 157
- device driver, 14, 20, 81, 106
- device file, 63, 124, 139, 255
- df, 370
- DFS, 212
- dialog box, 70
- dictionary, 128
- diff, 351, 370
- diff3, 371
- differences, 297
- directories
 - in root, 124
- directors, 239, 242
- directory, 35, 358, 386, 389
 - home, 20
- directory tree, 41, 71, 123
- dirname, 372
- disk drive, 91, 106
 - floppy disk drive, 19
- disk formatting, 134
- disk image, 59, 369
- disk image file, 50, 53
- diskette, 105, 139, 374, 389
- DISPLAY, 155
- display, 389
- dist, 124
- Distributed Computing Environment, 212
- distributed development, 3
- Distributed File System, 212
- distribution, 69, 78
 - diskette package, 79

- MCC, 70
 - Slackware, 70
 - SLS, 70
 - DLink, 77
 - DNS, 254, 333
 - DNS and Bind, 202
 - doc string, 308
 - documentation, 129, 287
 - domain, 198, 239
 - domain name, 200, 411
 - DontZap, 157
 - DOS, 16, 40, 49, 50, 60, 93, 97, 212, 213, 218, 221, 267, 280, 287
 - boot diskette, 50
 - directory, 16
 - security, 41
 - DOS emulator, 49, 50, 61
 - boot image, 51
 - booting, 51
 - configuration, 50, 54
 - configuration file, 52
 - debug settings, 55
 - DOS partition, 52
 - ending, 61
 - help, 61
 - options, 60
 - DOS file system, 39, 41, 42, 54
 - DOS partition, 41, 50, 52, 54, 72, 95
 - DOS version, 49
 - dot clock, See clock
 - DPMI, 49, 58
 - drag and drop, 153, 335
 - drawing programs, 270
 - drive letter, 52, 53
 - driver, 51, 77, 81, 90, 103, 106
 - appendfile, 242
 - conflict, 90
 - keyboard, 157
 - mouse, 158
 - pipe, 242, 244
 - DSYSV, 350
 - du, 372
 - dual access, 54
 - DVI format, 275
 - dynamic binding, 308
 - dynamic IP address, 194
 - dynamic linker, 410
- E—
- e-mail, 182, 183, 231, 348, 387
 - address, 239
 - e2fsck, 39
 - echo, 373
 - ed, 373
 - ediff, 297
 - edit-options, 317
 - Editieren files, 397
 - editor, 267, 287, 406
 - editor functions, 312
 - editres, 172
 - efficiency, 64
 - egrep, 373
 - EISA bus, 76
 - ELF, 62
 - elk, 328
 - elvis, 266
 - Emacs, 7, 71, 125, 314, 339, 341, 343, 347, 348
 - basic installation, 288
 - color, 317
 - configuration, 314
 - configuration file, 296
 - documentation, 292
 - editor, 268
 - extension, 320
 - font, 317
 - help, 292
 - invocation, 289
 - Lisp, 287
 - manual, 294

- memory requirements, 288
- menu, 324
- modes, 295
- programs, 296
- reference card, 291
- tutorial, 292
- version 19, 269
- Emacs Editor, 8
- Emacs Lisp, 288, 295, 298
- Emacs Lisp Code Directory, 298
- Emboss, 269
- EMS, 49, 58
- emufs.sys, 52-54
- emulator
 - DOS, 49, 61
 - iBCS2, 61
 - IBM 3270, 67
 - Macintosh, 67
 - mainframe, 67
 - Windows, 61
- encode, 406
- enhanced shells, 21
- entertainment, 129
- env, 373
- Environment, 22
- environment variable, 22, 46, 128, 155, 169, 247, 373
- Epson, 112
- Erlangen, 73
- error message, 92, 108
- <Esc>, 317
- escape, 317
- ESQL-C, 280
- etags, 299
- etc, 99, 101, 124, 140
- etc/exports, 213
- etc/fstab, 99-101
- etc/group, 130
- etc/hosts, 201
- etc/hosts.equiv, 231
- etc/hosts.lpd, 109
- etc/inetd.conf, 225
- etc/init.d, 125
- etc/inittab, 37, 120
- etc/Isode, 125
- etc/issue, 133
- etc/keytables, 125
- etc/lilo, 263
- etc/lilo.conf, 94, 103
- etc/lilo/install, 104
- etc/login.defs, 101
- etc/magic, 374
- etc/motd, 133
- etc/passwd, 124, 130
- etc/ppp, 125
- etc/printcap, 109, 110, 112
- etc/rc.d, 125
- etc/rc?.d, 125
- etc/resolv.conf, 202
- etc/services, 225
- etc/shells, 132
- etc/skel, 125
- etc/syslog.conf, 108
- etc/syslogd.reload, 109
- etc/X11, 125
- eth0, 96
- eth1, 96
- Ethernet, 9, 103, 106, 181, 183, 185, 186, 188, 189, 192, 198
 - address, 188
- Ethernet adapter, 96
- Ethernet address, 186
- Ethernet board, 81
- Ethershare, 220
- eval-expression, 317
- evaluation, 303, 317
- execution speed, 146
- execution time, 404
- expanded memory, 49
- exports, 125
- expr, 373
- expression, 303, 373
- ext, 38
- ext2 file system, 134

extended file system, 38
 Extended2 file system, 38, 86
 extension, 185
 extension language, 328
 eXternal Data Representation,
 210
 ez (text processor), 278

—F—

face, 318
 false, 374
 FAQ, 227
 sources, 260
 fax, 114
 display, 115
 incoming, 114
 script, 115
 faxadmin, 114
 Faxlog, 116
 fdformat, 134, 139, 374
 fdisk, 84, 87, 94, 97
 fgrep, 374
 field, 367
 file, 156, 374
 extension, 316
 File Allocation Table, 16
 file compression, 348, 379
 file descriptors, 24
 file formats, 255
 file hierarchy, 15
 file manager, 265
 File Name Expansion, 25
 file server, 213
 file system, 2, 15, 16, 18, 38,
 50, 52, 54, 78, 82, 84,
 86, 87, 91, 94, 97, 99,
 103, 123, 126, 134,
 138, 370, 390, 409,
 410, 412
 creating, 86
 DOS, 41, 54
 extended, 38
 Extended2, 38, 86
 ISO9660, 39
 management, 134
 MINIX, 38
 proc, 39
 repair, 39, 135
 virtual, 18
 Xia, 39
 file system check, 134
 file system standard, 125, 129,
 130, 409
 file systems, 121
 DOS, 39
 OS/2, 39
 System V, 39
 file transfer, 377
 file tree, 11
 file type, 374, 410
 filter, 111–113
 find, 350, 375
 Finder, 297
 finger, 377
 fingerd, 223
 Finland, 72
 flag, 156, 163
 Flagship, 279
 flicker, 163, 165
 floating point, 301
 floppy disk, 88, 105
 floppy disk drive, 59, 409
 floppy streamer, 117
 FM Synth, 107
 font, 174, 288, 317
 directories, 156
 paths, 157
 server, 157
 font lock mode, 295
 fonts, 295
 FOR-loops, 30
 forcepaths, 240
 Foreground (class), 172
 foreign modules, 327
 form, 303

- format, 139, 312
 - formatting, 134, 374
 - Forschungszentrum Informatik (FZI), 279
 - Forth, 325
 - Fortran, 325, 327
 - forward-char, 312
 - forward-word, 312
 - Foxpro, 278
 - frame, 289
 - frame (datatype), 302
 - frame (monitor), 164
 - frame grabber, 77
 - free, 87
 - free software, 21, 69
 - Free Software Foundation, 7, 44
 - Freedom Desktop, 265
 - freeware, 144, 280, 325
 - freeware archives, 4
 - frequently asked questions, 227
 - Fresco, 153
 - fsck, 134
 - FSF, 7, 44, 132, 133, 200, 287, 294, 300, 326, 338, 349, 377
 - FTP, 73, 125, 182, 183, 295, 409
 - server, 348
 - FTP daemon, 411
 - FTP mail server, 74
 - FTP server, 2, 9, 69, 72, 76, 97, 348
 - ftp.aud.alcatel.com, 335
 - ftp.ifi.uio.no, 285
 - ftp.rrzn.uni-hannover.de, 344
 - ftpd, 132, 223
 - func-menu, 299, 320
 - function call, 303
 - function invocation, 330
 - function keys, 290, 323
 - function names, 292, 299
 - function-key-map, 323
 - functional language, 314
 - functions, 301, 309, 314, 366
 - fundamental-mode, 316
 - Future Domain, 76
 - fvwm, 38, 149, 173
 - fvwm.goodstuff, 179
 - fvwmrc, 149
- G—
- G3 file, 114
 - G3 format, 115
 - games, 71, 129, 256
 - gated, 223
 - gateway, 103, 189, 244
 - gawk, 338
 - GCC, 326
 - gcc, 326, 344, 378
 - gdb, 344
 - General Public License, 7
 - generic VGA server, 76
 - generic window manager, 150
 - getty, 114, 409
 - getty process, 120
 - Ghostsript, 9, 112, 115, 215, 272
 - global settings, 314
 - global variable, 306, 317
 - global-set-key, 322
 - globbing, 32
 - GMD, 327
 - HTTP server, 253
 - Gmelch, Matthias, 74
 - GNAT, 326
 - GNU
 - AWK utility, 258
 - C, 326
 - C compiler, 258
 - debugger, 347
 - make, 341
 - utilities, 295
 - GNU Ada compiler, 128

- GNU C, 4
 - GNU C compiler, 5, 135
 - GNU C/C++, 10
 - GNU chess, 227, 284
 - GNU compiler, 2
 - GNU Emacs, 258, 287
 - editor, 267
 - See Emacs
 - GNU project, 9, 44, 47
 - GNU tar, 47
 - GNU utilities, 71
 - GoodStuff, 149, 179
 - Gopher, 249
 - goto-char, 312
 - GPL, 7
 - gr-latin1.map, 102
 - graphic board, 154
 - graphic conversion, 115
 - graphic formats, 269
 - graphic programs, 57
 - graphical front end, 340
 - graphical user interface, 9, 69, 75, 78, 100
 - see GUI
 - graphicmode, 49
 - graphics
 - color, 272
 - conversion, 269
 - view, 236
 - Gravis UltraSound, 77, 107
 - greeting, 133
 - greeting message, 410
 - grep, 350, 351, 378
 - groff, 257, 274, 378
 - group, 124, 359, 379
 - group membership, 358
 - guard time, 164
 - GUI, 143, 147, 150, 151, 334, 335, 337
 - Motif, 151
 - OpenLook, 151
 - standard, 151
 - SUIT, 152
 - toolkit, 143
 - widget set, 150
 - gwm, 150
 - gzip, 48, 80, 135, 348, 379
- H—
- hard disk, 15, 75, 81, 88, 107, 409
 - copying to, 87
 - direct access, 59
 - hard disk image, 53
 - hard link, 17
 - hardware, 14, 50, 74, 80
 - bus system, 76
 - hard disk, 75
 - keyboard, 157
 - monitor, 159, 162
 - mother board, 76
 - mouse, 158
 - network, 181
 - network boards, 77
 - RAM, 75
 - SCSI, 76
 - supported, 76
 - video board, 92, 160, 162
 - Hardware HOWTO, 76
 - hardware independence, 144
 - head, 380
 - header files, 136, 350
 - help, 261, 278, 353, 407
 - Hewlett Packard, 151
 - high memory, 49
 - High Sierra, 39
 - high-level function, 328
 - history function, 20
 - history scrolling, 45
 - hlt, 106
 - home, 89, 126
 - home directory, 15, 20, 124, 126, 132, 231, 242, 244, 347, 412
 - home file system, 84

- hook, 320
 - horizontal cycle, 164
 - horizontal synchronization, 159
 - See horizontal timing
 - horizontal timing, 163, 165
 - host adapter, 107
 - Future Domain, 107
 - PAS 16, 107
 - T128, 107
 - host address, 184, 185
 - hostname, 198, 201–203, 239, 380
 - HOWTO, 76, 97
 - HP, 77, 151
 - HP Deskjet, 113
 - HP48, 64
 - HPGL, 270
 - HSF, 163
 - HTML3, 253
 - HTTP server, 253
 - httpd, 223
 - Hughes, David J., 279
 - Hurd, 8
 - Hypercard, 337
 - hypertext, 268
 - Hypertext Markup Language (HTML), 251
 - Hypertext Transfer Protocol (HTTP), 251
 - hyphenation, 317
- |—
- i-node, 16, 17
 - i-node table, 123
 - I/O address, 103
 - I/O ports, 58, 96, 106
 - iBCS, 62
 - iBCS2, 265
 - iBCS2 emulator, 6, 61, 278
 - IBM, 151, 276
 - ICCC, 148
 - ICMP, 254
 - icon bar, 149, 179
 - ICS, 227
 - id, 381
 - IDE, 94
 - IDE interface, 75
 - identd, 223
 - IDL, 153
 - idle loop, 106
 - idraw, 270
 - IEEE, 2, 7
 - if, 305
 - ifconfig, 189, 190, 199
 - ignore-errors, 320
 - image file, 79, 94, 105, 369
 - IMAP2, 234, 236
 - imapd, 223
 - INBOX, 235
 - inbox-path, 235
 - include files, 129
 - inconsistencies, 123, 135
 - incremental search, 292
 - indentation, 288, 301, 307
 - inetd.conf, 124
 - Info, 306, 309, 314, 316
 - Info files, 300
 - Info mode, 318
 - Info system, 129, 342
 - Ingres, 4, 278, 279
 - init, 96, 114, 121, 125
 - initialization, 314
 - inittab, 114
 - INN, 248
 - input filter, 110, 112
 - insert, 312
 - insets, 278
 - insmod, 62
 - install, 89, 126
 - install file, 349
 - installation, 78
 - directories, 79
 - directory, 349
 - file system, 86

- hard disk space, 75
 - minimum hard disk, 75
 - of packages, 130
 - packages, 89
 - remote, 79
 - target, 94
 - installation disks, 138
 - installation mode, 88
 - installation package, 69
 - installation procedure, 78
 - installation program, 69, 75, 78, 87, 88, 90, 126
 - installed package, 126
 - installed programs, 126
 - integer, 301
 - integration, 146
 - interactive, 313
 - interactive interface editor, 152, 153
 - Interactive Use, 21
 - interface board, 103
 - interface builder, 63, 334–336
 - interface editor, 152
 - interlace, 163
 - international character sets, 111
 - Internet, 3, 4, 71, 72, 75, 181, 182, 228, 231, 240, 244, 245, 248, 249, 251, 255, 258, 262, 265
 - connection, 240
 - direct, 238
 - no direct connection, 244
 - services, 227
 - videos, 273
 - Internet Chess Server, 227
 - Internet level, 183
 - Internet News, 244
 - Internet Protocol (IP), 183, 184
 - Internet Relay Chat, 248
 - Internet superserver, 225
 - interpreter, 288, 300, 302, 329, 334, 335, 338
 - interprocess communication, 12
 - Interrupt, 96
 - interrupt, 14, 58, 81, 103
 - InterViews, 153, 270
 - intrinsic, 151
 - IP, 200, 201
 - IP address, 88, 183–189, 191, 194, 196, 199, 240, 410
 - IP packet, 186, 189, 213
 - IPC, 12
 - ipop2d, 224
 - ipop3d, 224
 - IPX, 60
 - IRC, 248
 - Linux channel, 249
 - irc, 249
 - ISA bus, 76
 - ISDN boards, 6, 77
 - isdn4linux, 77
 - ISO 9660, 39
 - ISO Latin-1, 102
 - ISO/OSI model, 220
 - ISO/OSI Reference Model, 182
 - ISO/OSI stack, 220
 - iteration, 330
- J—
- job, 354, 355
 - joe, 267
 - join, 249, 381
 - Just Logic Technologies, 280
- K—
- Karlsruhe, Germany, 279
 - Kerberos, 232
 - kernel, 5, 14, 39, 69, 72, 77, 78, 82, 91, 96, 103,

- 116, 117, 126, 130, 138, 141, 256
 - compilation, 102–104
 - compression, 104
 - configuration, 103
 - configuration with rdev, 105
 - image, 104
 - parameters, 106
 - source code, 102
 - kernel backup, 96
 - kernel decompression, 81
 - kernel image, 92, 96, 104
 - backup, 95
 - old, 96
 - kernel image file, 95, 123
 - kernel mode, 15
 - kernel parameters, 91, 96
 - kernel process, 15
 - key bindings, 289, 291, 293, 313, 322
 - key combination, 307
 - key configurations, 291
 - keyboard, 55, 102, 156
 - action, 176, 177
 - German, 102
 - layout, 102, 125, 167, 307, 314, 412
 - layout configuration, 166
 - Keyboard Entry, 29
 - keyboard table, 70
 - keymap, 102, 289, 294
 - keymap (datatype), 302
 - kill, 109, 381
 - klogd, 108
 - Knuth, Donald E., 274
 - Korn shell, 20, 44
 - ksh, 382
- L—
- Lamport, Leslie, 275
 - LAN, 181, 200
 - LAN manager, 181
 - client configuration, 217
 - server installation, 213
 - LAN manager server, 213
 - language, 325
 - last, 382
 - L^AT_EX, 275
 - ld, 383
 - ldconfig, 137
 - ldd, 383
 - lemacs, 269
 - let, 305, 308
 - level, TCP/IP, 183
 - lex, 383
 - lib, 126, 136
 - library, 15, 126, 129, 139, 150, 353, 383
 - paths, 349
 - shared, 15
 - libXaw3D, 151
 - LILO, 40, 80, 81, 90, 93, 97, 106, 120, 124, 141, 263, 410, 412
 - configuration file, 96
 - FAQ, 97
 - removal, 97
 - uninstall, 97
 - User's Guide, 97
 - LILOXinstallation, 412
 - index, 330
 - line number, 320
 - link, 17, 125, 137, 384
 - deleting, 137
 - hard, 17
 - symbolic, 17
 - link counter, 17
 - linker, 383
 - linking, 346
 - links, 90
 - Linux
 - T_EX, 275
 - C library, 350
 - graphic programs, 269
 - history, 1

- kernel documentation, 263
- mailing lists, 261
- manuals, 262
- nroff/groff, 274
- pronunciation, 6
- symbol, 351
- text editors, 266
- X terminal, 143
- Linux directories, 123
- Linux distributions, 69
- Linux Documentation Project (LDP), 262
- Linux features (overview), 9
- Linux file system, 38, 52, 54, 84
- Linux File System Standard, 123
- Linux IRC channel, 249
- Linux Loader, 81, 90, 93, 99, 104, 106, 141
- Linux Loader, See LILO
- Linux native, 89
- Linux native partition, 84
- Linux Software Map, 4
- Linux source code, 91
- Linux swap, 87
- Linux swap partition, 84
- Linux Universe, 5
- Linux X server, 78
- Lisp, 150, 269, 325, 328
 - Code Repository, 322
 - condition, 303
 - evaluation, 303
 - form, 303, 304, 306
 - interaction, 306
 - interpreter, 300, 302
 - list, 301, 303, 304
 - macro, 315
 - manual, 300
 - property list, 301
 - style, 314
 - symbol, 301
 - variable, 306
- lisp-dir-apropos, 322
- list, 249, 301, 303, 304, 311, 329
- list element, 310
- list functions, 310
- list-faces-display, 318
- listp, 310
- lists, 243
- ln, 137, 384
- load, 320
- load-path, 316
- loadable module, 62
- loadable modules, 43
- loadkeys, 102
- local, 126
- local area network (LAN), 181
- Local Bus, 76
- local variables, 28, 308
- log files, 109, 128, 134, 400
- login, 20, 101, 133, 395, 402, 407, 410
- login delay, 101
- login prompt, 83, 92, 101, 120, 410
- login shell, 20, 131, 411
- login statistics, 382
- login.defs, 101
- look, 128
- look and feel, 147–151, 153, 170, 174, 178
- loopback, 185, 189
- loopback address, 185
- loopback device, 185
- loose coupling, 145
- lost+found, 39, 126
- low-level formatting, 134
- lpc, 384
- lpd, 34, 224
- lpq, 385
- lpr, 34, 112, 385
- lprm, 386
- lredir, 52–54

ls, 17, 18, 47, 386, 409
 LST, 138
 Lucid, 269

—M—

m4, 349, 387
 M4 preprocessor, 173
 MacDonald, Peter, 70
 MacDraw, 272
 Mach chipset, 76
 Macintosh, 337
 Macintosh emulator, 67
 macro, 302, 387
 macro processor, 349
 macros, 303, 328, 366
 magnetic tape, 362
 mail, 128, 205, 387
 multimedia, 278
 mail order, 74
 mailboxes, 3, 74, 236
 mailing lists, 243, 261
 mainframe, 11, 12, 67
 major modes, 288, 295
 major number, 19
 major version, 136
 major version number, 140
 vger.rutgers.edu, 261
 make, 62, 96, 104, 346, 349, 387
 rules, 342
 make clean, 91, 104
 make config, 91, 103
 make dep, 91, 104
 make directory, 35
 make disk, 105
 make zImage, 91
 make zlilo, 104
 make-face, 318
 make-string, 312
 makefile, 103, 104, 335, 341, 342, 346, 349, 387
 makewhatis, 256

man, 42, 255, 388
 MANPATH, 257, 388
 manual, 72
 Manual pages, 35, 129, 130, 350, 388
 map files, 95, 124
 Maple V, 281
 mark, 288
 marking, 322
 master boot record (MBR), 94, 97
 match-inet-addr, 240
 match-inet-hosts, 241
 mathematics library, 356
 MCC, 70
 mcopy, 40
 mdir, 40
 memory, 72, 134
 memory management, 13, 14, 58
 memory requirements, 100, 103
 menu, 175
 menu bar, 290
 msg, 389
 message, 94, 109, 133, 389, 408
 message file, 94
 message of the day, 133, 410
 <meta>, 290
 meta-characters, 32
 <Meta-Esc>, 317
 <Meta-Return>, 307
 <Meta-x>, 307, 313
 <Meta-x> compile, 346
 <Meta-x> next-error, 346
 <Meta-x> shell, 348
 Metacard, 337
 metacharacters, 26
 mformat, 134
 mgetty, 114
 Microsoft Windows, 40, 61, 152, 280

- X server, 154
 - MIDI, 44
 - MIME, 235, 236
 - minibuffer, 313, 317
 - minimal system, 104
 - minimum configuration, 72
 - minimum Linux system, 78
 - MINIX, 1, 2, 38
 - MINIX file system, 38, 139
 - minor modes, 295
 - minor number, 19
 - minor version, 136
 - mirror servers, 73
 - MIT, 71, 72, 143, 147, 232
 - Mitsumi CD-ROM, 107
 - mixer, 44
 - mkdir, 389
 - mkfs, 86
 - mknod, 116
 - mkswap, 87
 - mnt, 126
 - MOCKA, 327
 - mode, 288, 321
 - command/input, 266
 - mode lines, 161, 162
 - modem, 56, 90, 114, 191, 205
 - moderated newsgroup, 245
 - ModeShift, 158
 - modifier key, 156
 - modify file, 397
 - Modula-2, 325, 327, 339
 - compiler, 4
 - Modula-3, 325
 - monitor, 78, 156, 157, 159, 162
 - monitoring, 253
 - more, 389
 - more_hostnames, 239
 - Mosaic, 251
 - motd, 133
 - mother board, 76
 - Motif, 63, 147, 151, 152, 247, 251, 335
 - widget set, 149, 152, 268
 - window manager, 149
 - Motif library, 337
 - Motif widget set, 147, 334
 - mount, 16, 41, 42, 99, 126, 390
 - mount options, 100
 - mountd, 224
 - mountpoint, 100
 - mouse, 19, 56, 76, 90
 - mouse action, 176, 177
 - mouse driver, 156, 158
 - move, 393
 - moving a file, 35
 - MPEG video player, 273
 - MPU-401 UART, 107
 - MS-DOS, 389
 - MSQL, 279
 - MTools, 40, 411
 - mttools, 389
 - MUDs (Multi-User Dungeons), 227, 284
 - multi-user, 11, 12
 - multi-user mode, 123, 412
 - multicasting, 185
 - multimedia, 77, 278
 - multiple output, 110
 - multiprogramming, 100
 - multiserial adapter, 77
 - multitasking, 12, 78
 - multiuser/multitasking, 9
 - multivolume mode, 133
 - MuPAD, 280
 - mv, 393
 - mwm, 149, 152
 - MX records, 240, 241
- N—
- NAG, 202, 262
 - name, 380, 405
 - name extension, 292
 - name server, 411

- Nation, Robert, 149
 - nawk, 338
 - NCR, 76
 - NCR 53c810, 77
 - nesting depth, 307
 - Netscape, 251
 - network, 9, 12, 103, 109, 123, 145, 205, 412
 - address, 88, 184
 - administrator, 184
 - board, 77, 80, 90, 103, 181
 - generic, 77
 - classes, 184
 - daemons, 120, 130
 - file system, 211, 232
 - hardware, 181
 - hardware requirements, 181
 - interface, 412
 - level, 183
 - management, 253
 - mask, 88, 185, 187-189
 - parameters, 88
 - programming, 333
 - programs, 71
 - protocol, 146
 - services, 250, 410
 - traffic, 145
 - transparency, 61, 145, 146
 - Network Administration Guide, 202, 262
 - Network Information Center (NIC), 184
 - Network Information System, 211
 - netx, 60
 - neural networks, 281
 - new line, 42
 - New York University, 326
 - new_fax, 115
 - News, 224
 - news server, 245
 - news.answers, 228
 - newsgroups, 3, 245
 - Linux, 245
 - Newsreader, 246
 - NFS, 70, 79, 88, 121, 126, 130, 181, 210, 212, 218, 221, 224, 232, 409
 - NFS mount, 100
 - nfsd, 224
 - nic.funet.fi, 72, 138
 - nice, 393
 - nice levels, 13
 - nil, 302
 - NIS, 181
 - nm, 351, 393
 - nmbd, 224
 - NNTP, 245
 - nntpd, 224
 - NNTPSERVER, 247
 - no-hlt, 106
 - no387, 106
 - noauto, 99
 - nohup, 393
 - NORMAL, 89
 - North Carolina, University of, 72
 - notebook, 199
 - NoTrapSignal, 157
 - Novell, 49, 60, 77, 181
 - login, 60
 - nroff, 257, 274, 393
 - ntalkd, 224
 - nth, 311
 - number sysstems, 356
 - numberp, 310
- O—
- Oberon, 325
 - object embedding, 278
 - object files, 91, 104
 - object orientation, 148, 153, 154, 279, 280
 - object-oriented database, 279
 - ObjectBuilder, 335

- Objective C, 325
- ObjectLibrary, 335
- OBJS, 346
- OBST, 279
- ODBC, 280
- olvwm, 38, 149
- olwm, 149
- On-Line Manual, 257, 274
 - format, 256
 - pages, 255
 - path, 257
 - reference, 257
 - readable form, 257
- Online Documentation, 35
- Open Software Foundation,
 - 147, 149, 151, 152, 212
- open system, 144
- OpenLook, 130, 149, 151–153,
 - 281, 335
- virtual window manager,
 - 149
- window manager, 149
- openwin, 394
- OpenWindows Deskset, 151
- operating system, 1, 8, 13, 93,
 - 94, 96
- operating systems, 280
- or converting numbers, 356
- original Linux server, 72
- OS/2, 93, 280
- OS/2 boot manager, 93, 97
- OS/2 Warp, 251
- OSF/Motif, 4, 9, 151, 335
- other, 95
- Ousterhout, John, 328
- output, 35, 400
- output filter, 110, 112
- owners, 243, 359
- P—
- page feed, 110
- paging, 13
- pairing parentheses, 301
- Panasonic, 108
- pane, 340
- paragraph, 167
- parallel connection, 181
- parallel execution, 12
- parameters, 105, 106, 313
- ParcPlace, 335
- partition, 16, 41, 78, 94
 - root, 89
 - target, 89
- partition size, 87
- partition type, 84, 89
 - Linux native, 84
 - Linux swap, 84
 - type 82, 84
 - type 83, 84
- partitioning, 84
- PAS 16, 77
- Pascal, 327
- passwd, 124, 394, 410, 411
- password, 11, 36, 131, 132,
 - 394, 411
- patch, 351
- PATH, 22, 128
- path, 15, 156, 169, 171, 316,
 - 356, 395, 407
- pattern matching, 32
- PC-UNIX, 93
- PCI, 76
 - bus, 76
 - components, 77
 - consumer advice, 77
 - hardware, 77
 - HOWTO, 77
 - standards, 77
- pcnfsd, 218, 224
- pdipx, 60
- performance, 144, 146, 154,
 - 166
- peripheral devices, 77, 103
- permissions, 11, 17, 119, 358,
 - 405

- Picasso, 334
- pico, 234
- picture adjustment, 166
- picture processing, 269
- pine, 234
- pinerc, 236
- pipe operator, 25
- pixel, 163
- platform independence, 143, 342
- PLIP, 181, 199
- pocket adapter, 181
- POET, 280
- point, 288, 312
- Point-to-Point Protocol, 191
- pointer, 156, 310
- pointerColor, 172
- popup menu, 299
- port, 410, 411
- port assignment, 411
- port number, 200
- portability, 144, 337
- porting software, 7, 9
- porting to Linux, 72
- porting under UNIX, 7, 10
- porting UNIX, 6
- portmap, 210–212
- POSIX, 2, 7, 9
- POSIX standard, 7
- Postgres, 4, 279
- PostScript, 112, 215, 270, 271, 272, 274, 349
 - printer, 272
- ppm, 115
- PPP, 125, 181, 191, 194–197, 199
- PPP daemon, 198
- pppd, 194, 198, 224
- pr, 394
- predecessor version, 343
- predicate, 309
- prefix notation, 303
- primary group, 410
- primitive function, 302
- print server, 109, 213
- printcap, 110, 113
- printer, 59, 410
 - configuration, 109
 - interface, 110
 - queue, 109, 110, 112, 113, 128, 411
 - server, 110
- printer daemons, 34, 109
 - lpd, 34
- printer queue, 385, 386
- printer spooler, 384
- printing, 394
- priority, 13, 206, 354, 393
- privileged ports, 232
- Pro Audio Spectrum, 107
- probeonly, 162
- proc, 39, 100, 126
- proc file system, 39, 126
- process, 36, 126, 395
- process (datatype), 302
- process number, 121
- process state, 40
- process switching, 103
- process termination, 123
- processors, 58
 - 80286, 75
 - 80386, 12, 75
 - 80486, 103
- procmail, 236
- profile, 23
- profiling, 136
- progn, 305
- program, 129
- program manager, 265
- programming languages, 4, 9, 287, 325
- Prolog, 325, 328
- prompt, 23, 29, 46, 93, 94
- pronunciation, 6
- property, 170
- property list, 301

protocol daemon, 35
 protocol layer, 147
 prototype, 334, 337, 350
 prototyping, 338
 ps, 100, 395
 PS/2 mouse, 76
 PS1, 23
 pub/Linux/docs/LDP, 262
 public domain, 4, 144
 push, 316
 pushnew, 322
 puts, 330
 pwd, 395

—Q—

query language, 230
 quick, 88
 quote, 305
 quoting, 26

—R—

RAM, 75, 87, 100
 ramdisk, 91, 92, 105, 106, 139, 141
 Raster Image Processor (RIP), 215
 rawrite.exe, 80
 rc, 99, 102
 rc.0, 121
 rc.inet1, 121, 123
 rc.inet2, 121, 123
 rc.K, 121
 rc.local, 62
 rc.M, 121
 rc.S, 121
 rc.serial, 121
 rcp, 231, 395
 RCS, 341
 rdev, 91, 105, 138, 141
 read, 17, 29
 read-evaluate-print cycle, 302

read-only, 105, 106
 read/write, 106
 README, 80, 263, 349
 README.st, 107
 reboot, 90
 recompilation, 91
 recursive output, 35
 redirect standard output, 18
 redirection, 23, 155
 output, 155, 357
 refresh rate, 159, 165
 region, 288, 317
 regular expressions, 30, 32, 331
 Release 4 of System V, 7
 remote, 231
 debugging, 339
 host, 110
 installation, 79
 login, 402
 shell, 233
 Remote Procedure Calls (RPC), 210
 remounting, 84
 removable storage media, 16
 remove directory, 35, 296
 remove files, 35, 396
 rename, 35, 393
 repair of file system, 39
 Request for Comment, 182
 requests, 243
 resolution, 163, 164
 resolv+, 202
 resolver, 201, 241
 resources, 11
 definition, 170
 file, 171, 172
 manager, 148
 RESOURCE_MANAGER, 169
 <Return>, 307
 reverse, 311
 Revision Control System, 341

- RFC, 181, 182
 - via FTP or e-mail, 182
 - RGB color table, 156, 157
 - rhosts, 156, 231
 - RIP, 215
 - Ritchie, Dennis, 6
 - rlog, 343
 - rlogin, 155, 231, 395
 - rlogind, 224
 - rm, 396
 - rmail, 205
 - rmdir, 396
 - rn, 247
 - ro, 106
 - roadmap, 73
 - Rockridge Extensions, 39
 - root, 12, 83, 92, 101, 119, 124, 126, 127, 232, 338
 - directory, 123, 132
 - disk, 78, 79, 82
 - file system, 84, 91, 94, 97, 105, 106, 138, 141
 - name server, 204
 - partition, 89, 105, 106
 - Rose, Marshall T., 220
 - round robin, 13
 - routed, 224
 - router, 96, 103, 186–189, 192, 239, 244, 409
 - routine, 255
 - routing, 184, 187, 188, 190, 194, 196
 - routing table, 187–189, 191
 - RPC, 181
 - routines, 210
 - server, 210
 - rpc.mountd, 212
 - rpc.nfsd, 212
 - RR, 163
 - RSA, 253
 - rsh, 231, 396
 - rshd, 224
 - rstatd, 224
 - run, 347
 - run level, 120, 125, 397, 410
 - ruptime, 231
 - rusersd, 224
 - rw, 106
 - rwalld, 224
 - rwho, 231
 - rwhod, 224
 - RWTH Aachen, 253
- S—
- s-expression, 303
 - S3, 162
 - S3 chipset, 76
 - Samba, 213
 - save-excursion, 306
 - Saveplace, 296
 - SB16, 107
 - SB16 Midi, 108
 - sbin, 127
 - scan lines, 164, 165
 - scanner Generator, 383
 - SCCS, 343
 - scheduler, 13, 14, 102
 - strategies, 13
 - scheduling, 354
 - scheme editor, 280
 - SCO, 62
 - Open Desktop, 64
 - programs, 63
 - shared libraries, 63, 64
 - SCO software, 62
 - SCO UNIX, 278
 - scotty, 333
 - scotty, Tel interpreter, 253
 - scratch, 306
 - screen, 157
 - screen update frequency, 57
 - SCREEN_RESOURCES, 169
 - script, 104, 125, 267, 328, 332, 337, 338, 342, 349, 350
 - script language, 332

- scroll keys, 46
- SCSI, 5, 76, 80, 90, 107, 116
 - chip, 77
 - devices, 76
 - host adapter, 76, 80
 - streamer, 77
 - support, 103
- sDevice, 113
- sdiff, 397
- Seagate, 76
 - driver, 107
- search, 292, 375, 378, 399
 - path, 169
 - patterns, 32
- search forward, 312
- searching, 35, 36
- section, 167
 - Device, 160
 - Keyboard, 157
 - Monitor, 159
 - Pointer, 158
 - Screen, 162
 - serverFlags, 157
- sed, 266, 338, 397
- seek, 19
- segmentation fault, 14
- selection, 317
- selection list, 70
- sendfax, 115
- sendmail, 224
- sequence, 302
- sequencer, 44
- serial interface, 56, 114, 191, 196, 412
- Serial Line Internet Protocol, 191
- serial mouse, 76
- server daemon, 202
- ServerFlags, 156
- services, 124
- set, 329
- set-face-background, 318
- set-face-font, 318
- set-face-foreground, 318
- set-face-underline-p, 318
- setenv, 155
- setq, 305
- shadow, 411
- shadow file, 131, 410
- shadow password, 101
- shar, 348
- shared files, 129, 130
- shared libraries, 15, 63, 127, 410
- shell, 14, 15, 20, 21, 132, 338, 348, 356, 364, 399, 411
 - alternative, 44
 - archive, 348
 - bash, 44
 - Bourne Again, 21, 44
 - C, 46
 - Korn, 44
 - tcsh, 21, 46
- shell model, 14
- shell process, 14
- shell script, 265, 267, 350, 410
- shell variants, 21
- shlib, 63, 127
- short commands, 26
- shortcut, 346
- shutdown, 57, 123, 397, 412
- signal, 381
- Silicon Graphics, 149
- Simple User Interface Toolkit, 152
- Simula, 327
- Simulator for Neural Networks (SNNS), 281
- simultaneous execution, 13
- single-user mode, 75, 138, 412
- site-lisp, 314
- site-start.el, 314
- size, 372

- of files, 399
- Slackware, 5, 69, 70, 78, 133, 138, 409
- Slackware distribution, 78
- sleep, 398
- Slingshot, 153
- SLIP, 181, 189, 191, 192, 199, 409
- SLS, 70, 138
- smail, 224, 237
 - configuration, 238
- Smalltalk, 325
- smart_path, 244
- smart_transport, 244
- smarthost, 244
- smbd, 224
- SMC, 77
- SMTP, 239
- SNMP, 253, 333
- sockets, 249
- Softlanding Systems, 70
- software overview, 72
- software catalog, 4
- software list, 72
- Sony, 108
- sort, 398
- sound, 150
- sound board, 96, 103, 107
- Soundblaster, 77, 107, 108
- source code, 4, 71, 91, 96, 102-104, 130, 268, 285, 320, 335, 348
 - control system, 343
 - dependencies, 91
 - files, 91
 - kernel, 104
- source file, 387
- source medium, 87, 88
- sources of Linux, 72
- speakers, 59
- special characters, 158, 169
- special forms, 303
- specification language, 152, 153
- spell check, 128
- split, 365
- spool directory, 110
- spool files, 128
- SQL, 279, 280, 333
 - client/server database, 280
 - Just Logic, 280
- Stallman, Richard, 7, 287
- standard error channel, 24
- standard shells, 20
- standardization, 143, 147, 151, 152
- standards, 6
- Stanford University, 153, 270
- stepwise execution, 338, 347
- STONE project, 279
- storage, 370
- stream (datatype), 302
- stream editor, 266
- streamer, 77, 134
- streamer tape, 70, 88
- <Strg-H><I>, 258
- string, 302, 312, 328
- strings, 399
- strip, 399
- stty, 399
- stub, 126, 129
- su, 399
- subdirectory, 17
- subnetwork, 185
- subscription, 71
- substring, 312
- SUIT, 152
- Sun, 61, 130, 151, 153
- Sun audio device, 44
- Sun Microsystems, 149
- sunsite.unc.edu, 72
- SunView, 153
- superblock, 123
- superuser, 12, 127, 411
- support, 3

- supported hardware, 76
 - SVGA, 162
 - SVGA server, 154
 - Swallow, 179
 - swap file, 14, 75, 100, 101
 - swap partition, 14, 75, 84, 87, 100, 409
 - creation, 87
 - swap region, 78
 - swap space, 100
 - swapon, 87, 100
 - swapon -a, 101
 - switching modes, 162, 166
 - symbol, 301, 304, 350
 - symbol table, 393
 - symbolic address, 200, 241
 - symbolic link, 17, 124, 128, 136, 140
 - symbolic name, 240
 - sync, 93, 123
 - synchronization, 162, 163
 - synchronization pulse, 164
 - syntax error, 346
 - syslog, 411
 - syslog daemon, 35
 - debug mode, 109
 - syslog.conf, 109
 - syslogd, 108, 109
 - system, 405
 - administrator, 256
 - invocation, 255
 - system administration, 119, 124, 127, 130, 256, 338
 - system administrator, 11, 134
 - system crash, 134, 138
 - system files, 78, 99
 - system independence, 349
 - system information, 100
 - system program, 128
 - system repair, 138
 - system startup, 104, 125
 - system time, 367
 - System V, 7, 39, 206
 - System V Interface Definition, 7
 - system.fvwmrc, 173–178, 180
- T—
- t, 302
 - <Tab>, 346
 - tags, 299, 366
 - tail, 400
 - talk, 400
 - talkd, 224
 - Tannenbaum, Andrew, 1
 - tape drive, 107
 - tar, 47, 48, 133, 135, 136, 138, 348, 400
 - tar archives, 156
 - tar.Z, 348
 - target partition, 87, 89
 - task, 12
 - task-switching, 75
 - Taylor UUCP, 206, 207
 - taz, 348
 - Tcl, 279, 330, 331
 - application, 335
 - argument, 329
 - assignment, 329
 - associative array, 330
 - class browser, 333
 - control construct, 330
 - curly braces, 331
 - custom function, 331
 - database, 333
 - debugger, 333
 - declaration, 329
 - DNS, 333
 - extension, 332, 333, 335
 - function name, 333
 - global variable, 332
 - GUI, 334
 - initialization, 333
 - interpreter, 329, 332, 333
 - itcl, 333

- iteration, 330
- language, 329
- library, 329
- list, 329
- list element, 329
- network programming, 333
- object-oriented, 333
- quoting, 329
- regular expression, 331
- scotty, 333
- script, 332
- script language, 332
- SNMP, 333
- SQL, 333
- statement, 329
- string, 328
- substitution, 329, 331
- substring, 331
- TCP/IP socket, 333
- Tk, 334
- Tk widget set, 334
- UNIX system call, 333
- variable, 332
- variable evaluation, 329
- XF, 334
- Tcl interpreter
 - scotty, 253
 - tcl-dp, 249
- Tcl/Motif, 334
- Tcl/Tk, 249, 253, 279
- Tcl_CreateCommand, 333
- tcsh, 329
- TCP, 181, 183, 184, 191, 200, 254
- tcp wrapper, 410
- TCP/IP, 6, 9, 67, 70, 103, 146, 181, 184, 185, 199, 206, 213, 220, 239, 249, 279, 333, 411
 - levels, 182
- TCP/IP configuration files, 202
- TCP/IP stack, 213, 218
- tcpd, 224
- tcsh, 21, 45, 47
- teamwork, 343
- tee, 402
- telnet, 155, 183, 200, 230, 333, 402
- telnetd, 155, 225
- template
 - function/class, 268
- template file, 342
- temporary files, 127
 - deletion, 127
- termcap, 323
- termcap library, 411
- terminal, 108, 323, 411
- termination, 393, 397
- terminfo, 323
- test, 29, 402
- Tetris, 284
- T_EX, 130, 270, 274, 349
 - data files, 130, 293
- texinfo format, 293
- text, 312
- text conversion, 43
- text editing, 265
- text files, concatenation, 35
- text mode, 43, 162
- text replacement, 404
- text-mode, 316
- textual commands, 340
- tftpboot, 127
- tftpd, 225
- tgz, 348
- Thompson, Ken, 6
- time, 354, 367, 404
- timed, 225
- timer interrupt, 55
- timing, 162
- tin, 247
- title page, 110
- Tk, 334
- tkined, 253, 334
- tkinfo, 129, 258, 294

- tmp, 127, 134
 - Tool Command Language See Tcl, 328
 - toolkit, 147
 - Fresco, 153
 - XView, 153
 - Torvalds, Linus, 1-4
 - touch, 109, 404
 - tr, 111, 404
 - trackball, 76
 - transient-mark-mode, 317
 - translator, 327
 - Transmission Control
 - Protocol, 183
 - transmission speed, 190
 - transport, 239
 - local, 242, 243
 - pipe, 242
 - uux, 242
 - transputer boards, 77
 - trn, 247
 - troff, 405
 - true, 302, 405
 - trumpet, 247
 - trusted users, 231
 - tsx-11.mit.edu, 72, 74, 135, 136
 - Turbo Pascal, 49, 267, 327
 - twm, 148
 - type, 301, 309
 - typesetting, 274
- U—
- UDP, 181, 183, 200, 254
 - UIC, 154
 - UIL, 152
 - umask, 405
 - UMB, 49
 - umlauts, 102, 111, 112
 - umount, 41, 92
 - uname, 405
 - uncompress, 405
 - undo, 288
 - Unifix, 72
 - Uniform Resource Locator, 251
 - uniq, 406
 - University of California at Berkeley, 273, 279
 - University of Paderborn, 280
 - University of Stuttgart (Germany), 281
 - University of Virginia, 152
 - UNIX, 1, 2, 4, 6, 11, 16, 61, 280, 287
 - administration, 119
 - commands, 35
 - compress, 348
 - directory, 16
 - PC platforms, 280
 - porting, 6
 - standards, 9
 - structure, 11
 - system call, 333
 - System V, 151, 343
 - Unix to Unix Copy, 205
 - unpack, 348
 - update, 72, 119
 - C library, 136
 - upgrade, 119
 - upper memory blocks (UMB), 49
 - uptime, 406
 - URL, 251
 - USENET, 245
 - user, 78, 119, 124-127, 377, 407
 - User Datagram Protocol, 183
 - user definition, 411
 - user ID, 11, 16, 130, 381
 - User Interface Language, 152
 - user mode, 15
 - user process, 14
 - useradd, 130, 412
 - userdel, 131

- usr, 89, 99, 127, 132
 - usr/adainclude, 128
 - usr/adm, 128
 - usr/bin, 21, 128
 - usr/bin/deliver, 244
 - usr/bin/X11, 128
 - usr/dict, 128
 - usr/doc, 129
 - usr/etc, 129
 - usr/g++-include, 129
 - usr/games, 129
 - usr/include, 129, 350
 - usr/info, 129
 - usr/lib, 129, 136
 - usr/lib/aliases, 242
 - usr/lib/dosemu, 50
 - usr/lib/kbd/keytables, 125
 - usr/lib/keytables, 125
 - usr/lib/X11, 129
 - usr/lib/X11/xinit, 166
 - usr/local, 129
 - usr/local/man, 256, 257
 - usr/man, 130, 256, 257
 - usr/openwin, 130
 - usr/pkg, 130
 - usr/sbin, 130
 - usr/share, 130
 - usr/spool, 34
 - usr/src, 46, 130, 138, 263
 - usr/src/linux, 91, 96, 103, 104
 - usr/src/linux/arch/i386/boot, 91
 - usr/TeX, 130
 - usr/X11R6, 128
 - usr/X386, 128
 - utilities, 78
 - uucico, 206
 - UUCP, 71, 181, 195, 206, 209, 239, 240, 242
 - configuration, 207
 - connection, 240
 - log files, 209
 - spool directory, 242
 - uudecode, 406
 - uuencode, 406
 - uulog, 206
 - uuname, 206
 - uustat, 206
 - uux, 206
 - uuxqt, 206
- v—
- var, 99, 128
 - var/lib/dosemu, 50
 - var/lib/smail, 238
 - var/log, 109
 - var/spool/mail, 235, 242
 - variable declaration, 338
 - variables, 22, 28, 306, 338
 - VBA, 328
 - vector graphics, 270
 - verbose, 88
 - verification of blocks, 86
 - version, 315, 405
 - Version 1.0, 4
 - Version 11 (X Window System), 143
 - version management system, 342
 - version number, 5, 136, 140, 343
 - version upgrade, 15
 - vertical refresh, 159
 - vertical refresh rate, 159
 - vertical synchronization frequency, 165
 - vertical timing, 163, 165
 - vgaset, 166
 - vgetty, 114
 - vi, 266, 287, 406
 - video adapter, 57
 - video board, 76, 92, 156, 157, 160, 162
 - video mode, 106, 159, 162

copying, 166
 video player, 273
 View, 279
 vim, 266
 Virginia, University of, 152
 virtual connection, 183
 virtual console, 37, 50, 78
 virtual desktop, 149
 virtual drive, 60
 virtual file system, 18
 virtual memory, 100
 virtual screen, 149
 virtual terminals, 12
 virtual window manager, 38,
 149
 Visual Basic for Applications,
 328
 vmlinuz, 92, 95, 104
 vmlinuz.old, 95
 VMS, 287
 Volkerding, Patrick, 71
 VXP, 335

—W—

w, 173, 407
 WABI, 61
 WAIS, 250, 262
 WAM, 328
 WAN, 200, 253
 watchpoint, 339
 Waterloo Software, 281
 wc, 407
 WD, 77
 web.cs.ualberta.ca, 335
 welcome message, 83
 whatis, 230, 256, 407
 which, 407
 while, 305
 WHILE-loops, 31
 who, 407
 wide area network, 200
 widget, 147

attribute, 171
 attributes, 148
 library, 147
 resource, 172
 widget set, 148, 150, 334
 Motif, 152
 wildcards, 25, 171, 172
 window, 289
 window (datatype), 302
 window decoration, 148
 window manager, 130, 143
 configuration, 173
 generic, 150
 OpenLook, 149
 Windows API calls, 61
 Windows Application Binary
 Interface (WABI), 61
 Windows emulator, 61
 Windows for Workgroups, 218
 Windows NT, 1, 7
 WINE, 61
 WordPerfect, 49, 62, 63
 Wordstar, 267
 workstation, 1, 12
 write, 408
 WWW, 72, 251, 261, 335
 browser, 251
 clients, 251
 Wyse, 62

—X—

X 11
 editor, 267
 X client, 144, 145
 X Consortium, 143, 148
 X Inside, 154
 X protocol, 145
 X resource database, 148
 X server, 78, 129, 144, 145,
 154, 156
 MS-Windows, 154
 X Toolkit, 147, 148

-
- X Toolkit Intrinsics Library, 147
 - X Window System, 4, 6, 9, 75, 128, 247, 267, 269
 - features, 143
 - structure, 146
 - X-Designer, 63
 - X/Open Portability Guide, 7
 - X11, 9, 50, 57, 60, 61, 69, 75, 76, 128, 129, 143, 287, 289, 294, 344, 349
 - configuration, 125, 143
 - graphical environment, 268
 - R5, R6, 272
 - X11R1, 143
 - X11R5, 71
 - X11R6, 76, 128
 - xanim, 273
 - XAPPLERESDIR, 169
 - xarchie, 230
 - xargs, 408
 - xboard, 227, 284
 - Xconfig, 156, 162
 - xcoral, 267
 - Xdefaults, 125, 170, 173
 - xdm, 120
 - xdos, 60
 - XDR, 210
 - xdvi, 275, 276
 - Xenix, 39, 62
 - XF, 334
 - xf, 334
 - XF86Config, 125, 156
 - xfig, 270
 - XFILESEARCHPATH, 169
 - XFree86, 64, 144, 154
 - XFree86 HOWTO, 76
 - xgopher, 251
 - xhost, 155
 - Xia file system, 39
 - Xia, Frank, 39
 - xinfo, 258
 - xinit, 166
 - xinitrc, 166
 - xkeycaps, 166
 - Xlib, 146, 147, 151
 - xmkmf, 342, 349
 - Xmodmap, 125, 166, 167
 - xmodmap, 166
 - XMS, 58
 - xntpd, 225
 - xpaint, 272
 - XPCE, 328
 - xrdb, 169, 170
 - xrn, 247
 - XT, 75
 - Xt library, 147
 - XTeddy, 283
 - xterm, 56, 156, 169
 - xv, 269
 - XView, 70, 130, 151, 153
 - xvnews, 247
 - xxgdb, 340
- Y—
- YARD, 280
 - Yard Software, 280
 - Yggdrasil Computing, 71
- Z—
- zcat, 408
 - zImage, 91
 - Zircon, 334
 - zircon, 249
 - Zyxel, 114